МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ имени Низами

Л.М. Набиулина

ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Учебное пособие

Ташкент – 2016

Данное учебное пособие предназначено для обучения студентов высших педагогических образовательных учреждений направления специальности 5111000 – Профессиональное образование (Информатика и информационные технологии) учебной дисциплине «Объектно-ориентированные языки программирования». В нем представлены лабораторные работы по обучению основам программирования на языке C++ в среде Borland C++ Builder 6.0.

Учебное пособие рекомендовано студентам педагогических университетов и институтов, а также учителям школ, академических лицеев и профессиональных колледжей, слушателям институтов и факультетов повышения квалификации.

Рецензенты:

Мигранова Э.А. – кандидат технических наук

Алибеков С.А. – кандидат физико-математических наук.

Рекомендовано к печати Ученым советом Ташкентского государственного педагогического университета имени Низами (протокол №9 от 18.06.2015)

СОДЕРЖАНИЕ

ТЕМА 1. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

1. ЦЕЛЬ ЗАНЯТИЯ

Целью занятия является приобретение знаний об основных понятиях объектно-ориентированного программирования, а также современных языках программирования.

2. ОСНОВНЫЕ ПОНЯТИЯ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

С момента появления первых компьютеров началось бурное развитие языков программирования. Решение различных задач привело к появлению процедурного программирования (procedure-oriented programming). Его основная идея заключалась в разделении программы на ряд независимых процедур. В начале 60-х годов была предложена концепция структурного программирования (structured programming). Она предполагала создание удобночитаемых программ, характерными особенностями которых являются модульность, отказ от оператора безусловного перехода, ограниченное использование глобальных переменных. Основная идея структурного программирования соответствует принципу "Разделяй и властвуй" т.е. программа делится на ряд простых подзадач.

К концу 80-х годов стали очевидными некоторые недостатки структурного программирования. Небыло возможности работать с данными и действиями выполняемыми над ними, как с единым целым. Программисты обнаружили, что они постоянно переизобретают новые решения старых проблем. Постепенно назрела идея объектно-ориентированного программирования (object-oriented programming) и проектирования.

Объектно-ориентированное программирование (ООП) предоставляет технологию управления элементами любой сложности. Суть ООП состоит в том, чтобы обращаться с данными и процедурами, которые выполняют действия над этими данными, как с единым объектом, т.е. самодостаточным элементом, который в чём-то идентичен другим таким же объектам, но в то же время отличается от них определёнными уникальными свойствами.

Методы ООП позволяют перейти от алгоритмических моделей программ к объектным. При ООП пользователя в первую очередь заботят типы объектов, с которыми приходится иметь дело их программам, свойства этих объектов, а также то, как они взаимодействуют между собой и с другими пользователями.

В объектно-ориентированном программировании главной отправной точкой при проектировании программы является не процедура, не действие, а *объект*. Такой подход достаточно естественен, поскольку в реальном мире нас окружают именно объекты, взаимодействующие друг с другом.

Объектно-ориентированное программирование базируется на трех основных принципах: *наследование, инкапсуляция и полиморфизм*. Программа, построенная по этим принципам, - это не последовательность операторов, не некий жесткий алгоритм, а совокупность объектов и способов их взаимодействия. Обмен информацией между объектами происходит посредством *сообщений*.

Объекты

Назовем объектом понятие, абстракцию или любой предмет с четко очерченными границами, имеющий смысл в контексте рассматриваемой прикладной проблемы. Все объекты можно идентифицировать, между объектами можно установить отношение тождества (два Сидоровых Ивана – это разные люди, но они студенты одной группы).

Классы объектов

Классом называют особую структуру, которая может иметь в своем составе поля, методы и свойства. Класс выступает в качестве объектного типа данных, а объект – это конкретный *экземпляр* класса. Например, два Сидоровых Ивана принадлежат одному и тому же классу объектов, они - студенты группы ТКС-206. Именно с этим связана их одинаковость (одинаковый шифр группы, одно расписание занятий и т.д.).

Каждый конкретный класс имеет свои особенности поведения и характеристики, определяющие этот класс. Например,

Наивысший уровень – самый общий и самый простой, каждый последующий уровень более специфический и менее общий. На самом последнем уровне можно определить цвет, стиль заполнения, величину радиуса окружности и т.п.

Если характеристика уже однажды определена для более высокого уровня, то все уровни, расположенные ниже имеют ту же характеристику (если уж определена окружность, понятно, что вершин у нее нет).

Таким образом, классы-наследники могут наследовать характеристики классов-родителей.

Свойства

Свойства – перечень параметров объекта, которые определяют внешний вид и поведение объекта, выделяют уникальные особенности каждого экземпляра. К свойствам относятся: имя, тип, значение, цвет, размер и др. *Состояние* – совокупность всех свойств данного объекта.

Методы

Метод - это некоторое действие (операция), которое можно выполнять над данным объектом. В результате этого действия в объекте что-нибудь меняется (например, местоположение, цвет и др.). Другими словами можно еще сказать,

методом называется команда, которую может выполнять объект. Для каждого класса объектов имеется свой перечень методов, которые можно к нему применить или которые он может выполнить. Например, объект можно удалить с экрана, переместить в другое место.

События

Каждый объект способен реагировать на определенные события – это разновидность свойства объекта. При возникновении события производится его обработка.

События – сигналы, формируемые внешней средой, на которые объект должен отреагировать соответствующим образом.

Средой взаимодействия объектов являются *сообщения*, генерируемые в результате наступления различных *событий*.

События наступают в результате действий пользователя – перемещение курсора пользователя, нажатия кнопок мыши или клавиш на клавиатуре, а также в результате работы самих объектов. Для каждого объекта определено множество событий, на которые он может реагировать. Для конкретных экземпляров объекта могут быть определены *обработчики* каких-то из этих событий, которые и определяют реакцию данного экземпляра объекта.

Объект можно определить как совокупность свойств и методов, а также событий, на которые он может реагировать.

Внешнее управление объектом осуществляется через обработчик событий. Эти обработчики обращаются к методам и свойствам объекта.

Инкапсуляция

С одной стороны объект, обладает определенными свойствами, которые характеризуют его состояние в данный момент. С другой стороны над объектами возможны операции, которые приводят к изменению этих свойств. Доступ к изменению свойств осуществляется только с помощью методов, присущих данному классу объектов. Есть метод, данное свойство данного объекта можно изменить, нет метода – нельзя. Методы как бы «окружают» свойства объекта, говорят, что свойства «инкапсулированы» в объект. Для обеспечения инкапсуляции класс не должен позволять прямого доступа к своим данным. *Инкапсуляция* - механизм скрытия всех внутренних деталей объекта, не влияющих на его поведение.

Наследование

Классы-наследники могут наследовать характеристики классов-родителей. Т.е. один объект приобретает свойства другого объекта, добавляя к ним свойства, характерные только для него.

Наследование определяет отношение между классами: объекты класса-наследник обладают всеми свойствами и методами объектов класса-родитель и не должны их повторно реализовывать.

Класс Точка (родитель)		**Класс Окружность (наследник)**	
Свойства	**Методы**	**Свойства**	**Методы**
Координаты (x,y)	**Перемещение**	**Координаты центра (x, y)**	**Перемещение**

Цвет	Изменение цвета	Цвет	Изменение цвета
		Радиус	Изменение радиуса

Полиморфизм (имеющий много форм)

К объектам разных классов можно применять один и тот же метод, вот только действовать этот метод будет по-разному. Например, к большинству объектов в Windows&Office можно применять одни и те же методы: копирование, перемещение, переименование, удаление и т.п. Однако, механизмы реализации этих методов для разных классов (файл в Windows и документ Word) неодинаковы.

Полиморфизм – возможность использования одних и тех методов для объектов разных классов, только реализация этих методов будет *индивидуальной* для каждого класса.

3. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Пионером данного направления явился язык Смолток (Smalltalk), первоначально предназначенный для реализаций функций машинной графики. Работа над языком началась в 1970 г. в исследовательской лаборатории XEROX (США), а закончилась в 1980 г. окончательным вариантом интерпретатора Smalltalk-80. Данный язык оригинален тем, что его синтаксис очень компактен и базируется исключительно на понятии объекта. В нем отсутствуют операторы или данные, все, что входит в Смолток, является объектами, а объекты общаются друг с другом исключительно с помощью сообщений. В настоящее время версия VisualAge for Smalltalk активно развивается компанией IBM.

К наиболее распространенным современным языкам программирования относятся C++ и Java.

Язык C++ был разработан в начале 80-х гг. Бьярном Страуструпом в лаборатории Bell корпорации AT&T. Им была создана компактная компилирующая система, в основе которой лежал язык C, дополненный элементами языков BCPL, Simula-67 и Алгол-68. Более ранние версии языка были известны как «C с классами». В июле 1983 г. C++ был впервые использован за пределами исследовательской группы автора, однако тогда еще многие особенности языка не были придуманы. К 1990 г. была выпущена третья версия языка C++, стандартизированная американским государственным комитетом стандартов ANSI. В 1990 г. сотрудник корпорации Sun Д. Гослинг на основе расширения C++ разработал объектно-ориентированный язык Oak, основным достоинством которого было обеспечение сетевого взаимодействия различных по типу устройств. Новая интегрируемая в Internet версия языка получила название Java. С января 1995 г. Java получает распространение в Internet.

По определению автора, Java является простым объектно-ориентированным и архитектурно-нейтральным языком интерпретирующего типа, обеспечивающим надежность, безопасность и переносимость, обладает высокой производительностью, многопоточностью и динамичностью.

Синтаксис языков C++ и Java практически полностью совпадает. Принципиальным различием является то, что язык C++ компилируемый в машинный код, а Java — в платформо-независимый байт-код (каждая команда занимает один байт), этот байт-код может выполняться с помощью интерпретатора — виртуальной Java -машины (Java Virtual Machine), версии которой созданы сегодня для любых платформ. С точки зрения возможностей объектно-ориентируемых средств, Java имеет ряд преимуществ перед C++. Язык Java имеет более гибкую и мощную систему инкапсуляции информации. Механизм наследования, реализованный в Java, обязывает к более строгому подходу к программированию, что способствует надежности и читабельности кода. Язык C++ обладает сложной неадекватной и трудной для понимания системой наследования. Возможности динамического связывания объектов одинаково хорошо представлены в обоих языках, но синтаксическая избыточность C++ и здесь принуждает к выбору языка Java. Сегодня Java по популярности занимает второе место в мире после Бейсика.

Идеи ООП проникли во многие процедурные языки. Например, в состав интегрированной системы программирования Паскаль (корпорации Borland International), начиная с версии 5.5, входит специальная библиотека ООП Turbo Vision.

С середины 90-х гг. многие объектно-ориентированные языки реализуются как системы визуального программирования. Такие системы имеют интерфейс, позволяющий при составлении текста программы видеть те графические объекты, для которых она пишется. Отличительной особенностью этих систем является наличие в них среды разработки программ из готовых «строительных блоков», позволяющих создавать интерфейсную часть программного продукта в диалоговом режиме, практически без написания программных операций. Система берет на себя значительную часть работы по управлению компьютером, что делает возможным в простых случаях обходиться без особых знаний о деталях ее работы. Она сама пишет значительную часть текста программы: описания объектов, заголовки процедур и многое другое. Программисту остается только вписать необходимые строчки, определяющие индивидуальное поведение программы, которые система не в состоянии предвидеть. Но даже в этих случаях система сама указывает место для размещения таких строк. К объектно-ориентированным системам визуального проектирования относятся: Visual Basic, Delphi, C++ Builder, Visual C++. Это системы программирования самого высокого уровня.

VBA (Visual Basic for Application) является общей языковой платформой для приложений Microsoft Office (Excel, Word, Power Point и др.). VBA соблюдает основной синтаксис и правила программирования языков Бейсик-диалектов. VBA помогает довольно сильно расширить возможности приложений за счет написания макросов —программ, предназначенных для автоматизации выполнения многих операций. VBA позволяет создавать объекты управления графического интерфейса пользователя, задавать и изменять свойства объектов,

подключать к ним необходимый для конкретного случая программный код. С помощью VBA можно производить интеграцию между различными программными продуктами. Программы на языке VBA для приложений создаются двумя способами: в автоматическом режиме как результат построения клавишной макрокоманды или путем написания программного кода.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

ОТВЕТЬТЕ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем заключается основное отличие процедурного и объектно-ориентированного программирования?
2. Назовите основные принципы объектно-ориентированного программирования
3. Что такое объект?
4. Что такое класс?
5. В чем заключается наследование?
6. Что такое инкапсуляция и для чего она применяется?
7. В чем заключается полиморфизм операций?
8. Что такое метод?
9. Что такое свойство объекта, каким образом его можно изменять?
10. Что такое события? Каково назначение обработчика событий?
11. Каково назначение сообщений?
12. В чем заключается преимущества визуального программирования интерфейса?

ИЗУЧИТЕ СЛЕДУЮЩИЕ ТЕМЫ:

1. Основные этапы развития технологии программирования.
2. Основы процедурного программирования.
3. Основы функционального программирования.
4. Основы логического программирования.

ТЕМА 2. ПРОГРАММЫ НА ЯЗЫКЕ С++ ЛИНЕЙНОЙ СТРУКТУРЫ

1. ЦЕЛЬ ЗАНЯТИЯ

Целью занятия является приобретение студентами знаний и навыков работы с приложением Borland C++ Builder 6.0 и освоение основ программирования на языке C++: объявление переменных, ввод/вывод сообщений и значений переменных, составление программ с линейной структурой.

2. КРАТКОЕ ОПИСАНИЕ СРЕДЫ Borland C++ Builder 6

После запуска среды разработки окно загрузки сменяется несколькими окнами, за которыми просматривается рабочий стол компьютера. Все эти окна представляют собой интерфейс программы Borland C++ Builder 6. В терминологии программистов этот интерфейс называется ***средой быстрой разработки приложений RAD*** (Rapid Application Development). Такое название он получил за то, что создание программы в нем сводится к простому конструированию внешнего вида будущей программы из готовых кубиков, а большую часть стандартных и рутинных операций за человека выполняет компьютер.

Самое верхнее главное окно интерфейса имеет заголовок **C++Builder 6 - Projectl,** который отражает название среды разработки и имя нового проекта, из которого будет получена работающая программа.

Проектом называется вся группа программных файлов, которые необходимы для создания конечной исполняемой программы. Так, например, в состав проекта могут включаться файлы с текстами программ, файл ресурсов с рисунками курсоров и иконок (значков), звуковые файлы и т. п. Первоначально проект хранится в памяти компьютера, и для того чтобы сохранить его на диске, необходимо будет выполнить стандартные операции сохранения, создав при этом отдельную папку. Кроме того, интерфейс сам предложит сохранить проект, если вы решите выйти из программы или попытаетесь создать новый проект.

Рассмотрим главное окно интерфейса. На строке заголовка проекта находятся кнопки свертывания, восстановления и закрытия окна. Под заголовком размещается строка главного меню, которая предоставляет доступ ко всем функциям и командам среды разработки. Под главным меню располагаются быстрые кнопки, объединенные в группы по назначению. Они позволяют получить быстрый доступ к наиболее часто используемым командам.

Справа от быстрых кнопок расположена ***палитра визуальных компонентов VCL*** (Visual Component Library, библиотека визуальных компонентов). Это те самые объекты или программные компоненты, предназначенные для быстрого создания визуальных программ для Windows. Компоненты позволяют быстро создавать в программе различные программные

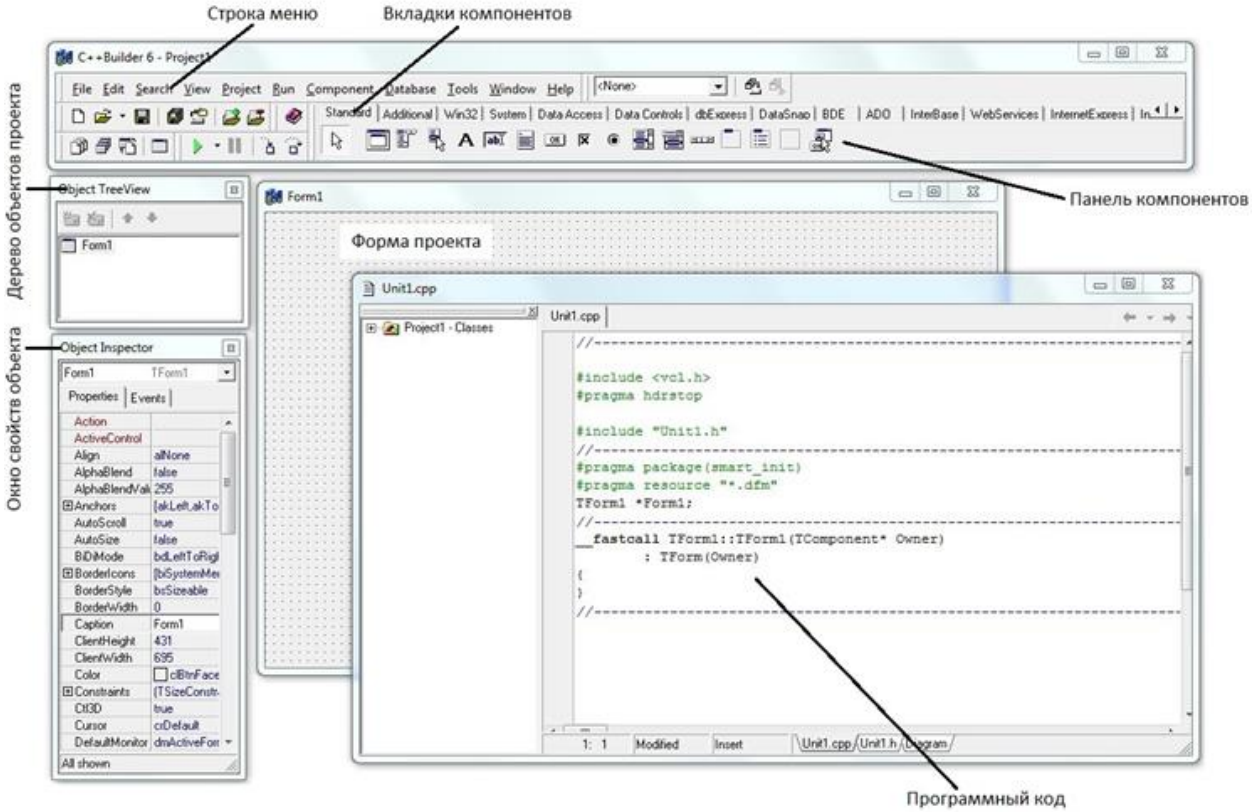
кнопки, рисунки, надписи, таймеры, календари и т. п. Палитра визуальных компонентов состоит из нескольких закладок, на которых располагаются компоненты, распределенные по группам. Именно с помощью этих компонентов вы будете быстро создавать новые программы.

В центре экрана располагается ***окно дизайнера форм***. Это окно будущей программы. Оно содержит строку заголовка, в котором отображается название формы Forml (созданное по умолчанию) и кнопки управления окном. На поле этого окна будут помещаться компоненты VCL в виде программных кнопок, надписей и других элементов будущей программы.

Под окном дизайнера форм располагается ***окно редактора кода*** с заголовком Unitl.cpp (также созданным по умолчанию), в котором производится набор и редактирование кода (текста) программы.

Следующее окно интерфейса располагается в левой нижней части экрана. Это ***окно инспектора объектов Object Inspector***. В этом окне производится настройка основных свойств визуальных компонентов. Расположение окна инспектора объектов в программе не фиксированное и при желании его можно переместить в ту часть рабочей области программы, которая для вас наиболее удобна. Для этого необходимо нажать левую кнопку мыши на строке заголовка окна и, удерживая ее, переместить окно.

Над окном инспектора объектов расположено ***окно просмотра объектов Object TreeView***. Оно отображает в виде дерева всю структуру проекта, состоящего из форм, кодов (текстов) программ и других ресурсов (файлов).



3. СОЗДАНИЕ КОНСОЛЬНОГО ПРИЛОЖЕНИЯ

Создадим с помощью C++Builder консольную программу Win32.

Консольное приложение (console application) Win32 представляет собой 32-разрядную программу, которая запускается в окне DOS под Windows.

1. Выберите в главном меню пункт File - New - Other. C++Builder откроет диалоговое окно New Items (окно репозитория). Данное диалоговое окно содержит предопределенные проекты, формы, диалоговые окна и другие объекты, которые вы можете добавить к существующему приложению или использовать для создания нового проекта.
2. На вкладке New выберите Console Wizard для создания нового консольного приложения и нажмите на кнопке OK.

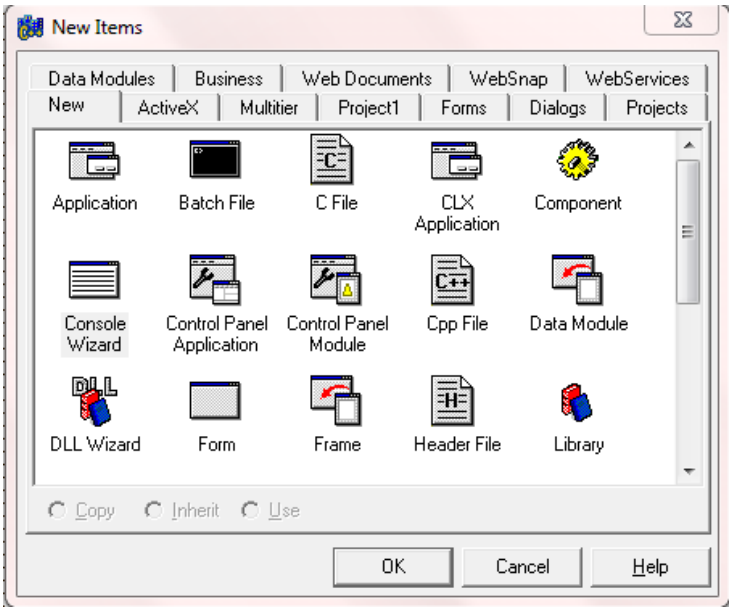
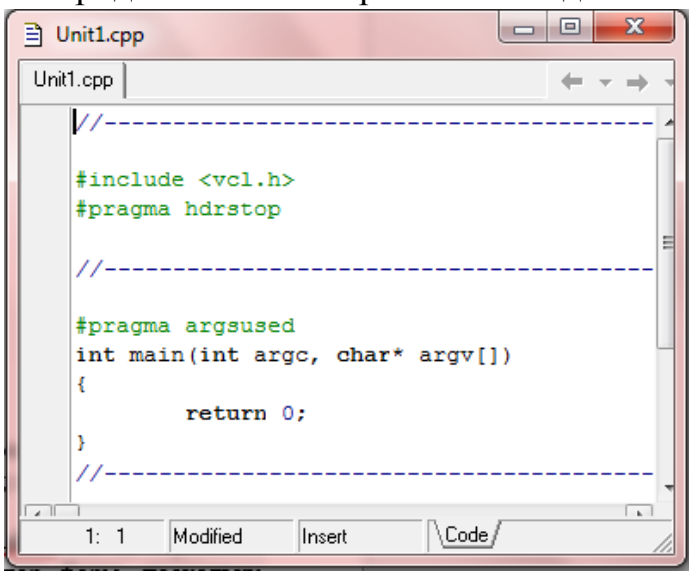


Рис. 1. Окно репозитория

C++Builder создаст проект и вызовет окно редактора кода, после чего вы можете начинать ввод текста программы. На рис. 2 показано окно редактора кода перед началом работы над новым консольным



приложением.

Вы заметите, что отсутствует редактор форм, поскольку консольные приложения не могут отображать формы (вообще говоря, это не совсем верно, но для нашего случая достаточно точно). Также обратите внимание, что окно инспектора объектов пусто. Компоненты можно помещать только в форму, поэтому при работе с консольными приложениями инспектор объектов не используется.

При написании консольных приложений вы можете закрыть инспектор объектов, чтобы освободить больше места для окна редактора кода. Для этого щелкните на кнопке Close, расположенной в строке заголовка окна инспектора объектов. Чтобы снова открыть инспектор объектов, нажмите клавишу F11 или выберите в главном меню пункт View | Object Inspector.

Обратившись к редактору кода, вы увидите, что в его окне находится следующий текст:

```
// - ---------------
«include <vcl.h>
#pragma hdrstop
// - ---------------
#pragma argsused
int main (int argc, char** argv[])
{
    return 0;
}
// - ---------------
```

Это программа на языке Си++, которая ничего не делает, но тем не менее написана правильно. Ее текст можно моментально изменить так, чтобы выполнялись какие-либо действия, но сначала обратим ваше внимание на строки, начинающиеся с //. Это строки комментариев, которые в данном случае введены только для визуального разделения текста программы. (Обычно строки комментариев используются для документирования текста.) C++Builder автоматически добавляет эти строки при создании нового консольного приложения. Обратите внимание, что единственный оператор в этой программе оканчивается точкой с запятой. Точка с запятой ставится в конце каждого оператора программы, написанной на Си++.

Обратите внимание также на открывающую и закрывающую скобки в программе. В Си++ блок кода начинается с открывающей фигурной скобки ({) и заканчивается закрывающей фигурной скобкой (}). Скобки используются для обозначения начала и конца блоков кода, связанных с циклами, функциями, операторами if, а также в других случаях. Данная программа очень проста, и в ней использован только один набор скобок.

Чтобы вывести на экран Hello World!, нам потребуется класс Си++, называемый iostream, поэтому нужно коротко рассказать об этом классе. (Вы еще не знаете о классах, но сейчас об этом не беспокойтесь.) Класс

iostream использует *потоки (streams)* для выполнения базового ввода и вывода, например, печати текста на экране или приема информации, вводимой пользователем. Поток cout используется для пересылки данных в стандартный выходной поток. В консольных приложениях стандартный выходной поток означает консоль, т.е. экран. Поток cin используется для приема с консоли данных, вводимых пользователем. В iostream включены два специальных оператора для помещения информации в поток или извлечения ее из потока. *Оператор вставки* (<<) используется для помещения данных в выходной поток, а *оператор извлечения* (>>) — для извлечения данных из входного потока. Для вывода информации на консоль вам следует написать

```
cout << "Do something!";
```

Это является указанием поместить текст Do something! в стандартный выходной поток. При выполнении данной строки программы текст будет выведен на экран.

cout предназначен для использования только в приложениях, работающих в консольном режиме. GUI-приложение Windows не имеет стандартного выходного потока (в GUI-приложениях все основано на графике), поэтому в таком приложении вывод из cout отправляется «в никуда». В стандартных Windows-программах для отображения текста на экране используется функция DrawText() или TextOut(). Программы, написанные с помощью C++Builder, также используют DrawText() или TextOut(), либо через Windows API, либо через класс TCanvas.

Перед тем, как использовать cout, необходимо сообщить компилятору, где находится описание (называемое *объявлением)* класса iostream. Объявление для iostream расположено в файле с именем IOSTREAM.H. Этот файл называется *заголовочным файлом (header file).*

Заголовочный файл (или, для краткости, *заголовок)* содержит объявления одного или нескольких классов.

Чтобы компилятор искал описание класса iostream в файле IOSTREAM.H, используйте директиву #include:

```
#include <iostream.h>
```

Теперь компилятор сможет найти класс iostream и поймет, что нужно делать, когда ему встретится оператор cout.

Если вы забудете включить заголовочный файл для класса или функции, на которые ссылается ваша программа, компилятор выдаст сообщение об ошибке/ Увидев сообщение типа Undefined symbol 'cout' (Не определен символ 'cout’), следует немедленно проверить, все ли необходимые заголовочные файлы включены в программу. Чтобы определить, в каком файле находится объявление класса или функции, щелкните на имени этого класса или функции и нажмите клавишу F1. Это приведет к запуску справочной системы Windows и на экране появится справка по объекту, на котором находится курсор. Где-то в верхней части справочного окна вы увидите ссылку на заголовочный файл, в котором

находится объявление функции или класса.

До того, как мы напишем консольную версию Hello World, упомянем еще об одной вещи. Класс iostream содержит специальные *манипуляторы (manipulators),* с помощью которых можно управлять процессом вывода. Сейчас мы будем иметь дело только с одним из них, а именно `endl` (end line - перевод строки), который вставляет символ новой строки в выходной поток. Мы используем `endl` для перехода на следующую строку после вывода текста на экран.

Теперь, когда вы имеете некоторое представление о классе iostream, мы можем написать программу Hello World как консольное приложение.

3. Отредактируйте текст, чтобы он выглядел как в примере 1.1.

Пример 1.1. HELLO.CPP

```
#include <vcl.h>
#include <iostream.h>     //добавить эту строку
#pragma hdrstop

//---------------------------------------------------

#pragma argsused
int main(int argc, char* argv[])
{
        cout <<"Hello World!"; //добавить эту строку

        return 0;
}
//---------------------------------------------------
```

В Си++ пробелы игнорируются. В большинстве случаев не имеет значения, где вы вставите пробел или новую строку. Очевидно только, что нельзя вставлять пробелы в середине ключевых слов или имен переменных; все остальное допустимо. Например, следующие строки кода эквивалентны:

```
int main(int argc, char** argv)
{
cout << "Hello World!";
return 0;
}
```

есть то же самое, что и

```
int main(int argc,char*1argv)
{
Cout<<"Hello Worid!";
return 0;
}
```

Очевидно, что первая запись читается легче и поэтому более предпочтительна.

4. Щелкните на кнопке Run оперативной панели. Программа будет откомпилирована и запущена. В момент запуска вы увидите окно DOS и слова Hello World!... которые через долю секунды исчезнут. Что же случилось? По достижении конца функции main() программа завершает свою работу и консольное окно сразу же закрывается. Чтобы этого не произошло, мы должны добавить к программе еще несколько строк. Стандартная библиотека Си содержит функцию getch(), предназначенную для ввода символа с клавиатуры. Ее мы и используем, чтобы предотвратить закрытие консольного окна.

5. Вновь отредактируйте текст программы так, чтобы он выглядел в соответствии с примером 1.2. Можете не добавлять строки комментариев. Не забудьте исключить номера строк.

Пример 1.2 HELLO.CPP (измененный)

```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>  //добавьте эту строку
#pragma hdrstop

//---------------------------------------------

#pragma argsused
int main(int argc, char* argv[])
{
cout <<"Hello World!";
 //Добавьте две строки
cout<< endl << "Для продолжения нажмите любую клавишую...";
getch();
      return 0;
}
```

На этот раз, когда после запуска программы на экране появится Hello World!, консольное окно останется открытым. Для завершения программы и закрытия окна нажмите любую клавишу на клавиатуре.

Для того, чтобы в консоли выводились русские символы, необходимо:

- добавить #include <windows.h>;
- в самом начале функции main добавить строчки SetConsoleOutputCP(1251) и SetConsoleCP(1251);
- в свойствах консольного окна (правый щелчок мыши на строке заголовка консоли – пункт Свойства, вкладка Шрифт) поставить шрифт Lucida Console по желанию увеличив размер шрифта.

4. ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ С++

4.1 Алфавит языка

Программа на языке С++ может содержать следующие символы:

• прописные, строчные латинские буквы A, B, C, …, x, y, z и знак подчеркивания;

• арабские цифры от 0 до 9;

• специальные знаки: “ { } , | [ ] ( ) + – / % * . \ ‘ : ? < = > ! & # ~ ; ^

• символы пробела, табуляции и перехода на новую строку.

Из символов алфавита формируют ключевые слова и идентификаторы. Ключевые слова – это зарезервированные слова, которые имеют специальное значение для компилятора и используются только в том смысле, в котором они определены (операторы языка, типы данных и т.п.).

Идентификатор – это имя программного объекта, представляющее собой совокупность букв, цифр и символа подчеркивания. Первый символ идентификатора – буква или знак подчеркивания, но не цифра. Идентификатор не может содержать пробел.

Прописные и строчные буквы в именах различаются, например, `ABC`, `abc`, `Abc` – три различных имени. Каждое имя (идентификатор) должно быть уникальным в пределах функции и не совпадать с ключевыми словами.

В тексте программы можно использовать комментарии. Если текст начинается с двух символов «косая черта» // и заканчивается символом перехода на новую строку или заключен между символами /* и */, то компилятор его игнорирует.

```
/*Комментарий может
 выглядеть так!*/
//А если вы используете такой способ,
//то каждая строка должна начинаться
//с двух символов «косая черта».
```

Комментарии удобно использовать как для пояснений к программе, так и для временного исключения фрагментов программы при отладке.

4.2 Данные

Для решения задачи в любой программе выполняется обработка каких-либо данных. Данные хранятся в памяти компьютера и могут быть самых различных типов: целые и вещественные числами, символы, строки, массивы. Типы данных определяют способ хранения чисел или символов в памяти компьютера. Они задают размер ячейки, в которую будет записано то или иное значение, определяя тем самым его максимальную величину или точность задания. Участок памяти (ячейка), в котором хранится значение определенного типа, называется переменной. У переменной есть имя (идентификатор) и значение. Имя служит для обращения к области памяти, в которой хранится значение. Во время выполнения программы значение переменной можно изменить.

Перед использованием любая переменная должна быть описана. Оператор описания переменных в языке C++ имеет вид:

```
тип имя_переменной;
```

или

```
тип список_переменных;
```

Типы языка C++ можно разделить на основные и составные.

К основным типам данных языка относят:

• char – символьный;
• int – целый;
• float – с плавающей точкой;
• double – двойной точности;
• bool – логический.

Для формирования других типов данных используют основные типы и так называемые спецификаторы. Типы данных, созданные на базе стандартных типов с использованием спецификаторов, называют *составными типами данных*.

В C++ определены четыре спецификатора типов данных:

• short – короткий;
• long – длинный;
• signed – знаковый;
• unsigned – беззнаковый.

Далее будут рассмотрены назначение и описание каждого типа.

4.2.1 Символьный тип

Данные типа char в памяти компьютера всегда занимают один байт. Это связано с тем, что обычно под величину символьного типа отводят столько памяти, сколько необходимо для хранения любого из 256 символов клавиатуры. Символьный тип может быть со знаком или без знака (табл. 2.1).

Таблица 2.1. Символьные типы данных

Тип	Диапазон	Размер
Char	–128…127	1 байт
unsigned char	0…255	1 байт
signed char	–128…127	1 байт

Пример описания символьных переменных:

```
char c, str; //Описаны две символьные переменные.
```

При работе с символьными данными нужно помнить, что если в выражении

встречается одиночный символ, он должен быть заключен в одинарные кавычки.

Например, `'a', 'b', '+', '3'`.

Последовательность символов, то есть строка, при использовании в выражениях

заключается в двойные кавычки: `''Hello!''`.

4.2.2 Целочисленный тип

Переменная типа `int` в памяти компьютера может занимать либо два, либо четыре байта. Это зависит от разрядности процессора.

Диапазоны значений целого типа представлены в таблице 2.2. По умолчанию все целые типы считаются знаковыми, т.е. спецификатор `signed` можно не указывать.

Таблица 2.2. Целые типы данных

Тип	Диапазон	Размер
`int`	– 32767 … 32767	4 байта
`unsigned int`	0 … 65535	4 байта
`signed int`	– 32767 … 32767	4 байта
`short int`	– 32767 … 32767	2 байта
`long int`	– 2147483647 2147483647	4 байта
`unsigned short int`	0 … 65535	2 байта
`signed short int`	– 32767 … 32767	2 байта
`long long int`	$-(2^{63}-1)$ … $(2^{63}-1)$	8 байт
`signed long int`	– 2147483647 2147483647	4 байта
`unsigned long int`	0 … 4294967295	4 байта
`unsigned long long int`	0 … $2^{64}-1$	8 байт

Пример описания целочисленных данных:

```
int a, b, c;
unsigned long int A, B, C;
```

4.2.3 Вещественный тип

Внутреннее представление вещественного числа в памяти компьютера отличается от представления целого числа. Число с плавающей точкой представлено в экспоненциальной форме mE±p, где m – мантисса (целое или дробное число с десятичной точкой), p – порядок (целое число). Для того чтобы перевести число в экспоненциальной форме к обычному представлению с фиксированной точкой, необходимо мантиссу умножить на десять в степени порядок. Например,

```
-6.42E+2 = -6.42.102 = -642,
 3.2E-6 = 3.2.10-6 =0.0000032
```

Обычно величины типа float занимают 4 байта, из которых один двоичный разряд отводится под знак, 8 разрядов под порядок и 23 под мантиссу. Поскольку старшая цифра мантиссы всегда равна 1, она не хранится.

Величины типа double занимают 8 байт, в них под порядок и мантиссу отводится 11 и 52 разряда соответственно. Длина мантиссы определяет

точность числа, а длина порядка его диапазон. Спецификатор типа long перед именем типа double указывает, что под величину отводится 10 байт.

Диапазоны значений вещественного типа представлены в таблице 2.3.

Таблица 2.3. Вещественные типы данных

Тип	Диапазон	Размер
`float`	3.4E-38 … 3.4E+38	4 байта
`double`	1.7E-308 … 1.7E+308	8 байт
`long double`	3.4E-4932 … 3.4E+4932	8 байт

Пример описания вещественных переменных:

```
double x1,x2,x3;
float X, Y, Z;
```

4.2.4 Логический тип

Переменная типа `bool` может принимать только два значения `true` (истина) или false (ложь). Любое значение не равное нулю интерпретируется как `true`, а при преобразовании к целому типу принимает значение равное 1. Значение `false` представлено в памяти как 0.

Пример описания данных логического типа:

```
bool F, T;
```

4.2.5 Тип void

Множество значений этого типа пусто. Он используется для определения функций, которые не возвращают значения, для указания пустого списка аргументов функции, как базовый тип для указателей и в операции приведения типов.

4.3 Константы

Константы - это величины, которые не изменяют своего значения в процессе выполнения программы. Оператор описания константы имеет вид:

```
const тип имя_константы = значение;
```

Константы в языке С++ могут быть целыми, вещественными, символьными или строковыми. Обычно компилятор определяет тип константы по внешнему виду, но существует возможность и явного указания типа, например,

```
const double pi=3.141592653589793
```

Кроме того, константа может быть определена с помощью специальной директивы:

```
#define PI 3.141592653589793
int main()
{…
```

4.4 Структурированные типы данных

Структурированный тип данных характеризуется множественностью

образующих его элементов. В С++ это массивы, строки, структуры и файлы.

Массив – совокупность данных одного и того же типа. Число элементов массива фиксируется при описании типа и в процессе выполнения программы не изменяется.

В общем виде массив можно описать так:

```
тип                                                    имя
[размерность_1][размерность_2]...[размерность_N];
```

Например,

```
float x[10];  //Описан массив из 10 целых чисел.
int a[3][4];  //Описан двумерный массив,
//матрица из 3-х строк и 4-х столбцов.
double b[2][3][2]; //Описан трехмерный массив.
```

Для доступа к элементу массива достаточно указать его порядковый номер, а если массив многомерный (например, таблица), то несколько номеров:

```
имя_массива[номер_1][номер_2]...[номер_N]
```

Например: `x[5], a[2][3], b[1][2][2]..`

Элементы массива в С++ нумеруются с нуля. Первый элемент, всегда имеет номер ноль, а номер последнего элемента на единицу меньше заданной при его описании размерности:

```
char C[5]; //Описан массив из 5 символов,
//нумерация от 0 до 4.
```

Строка – последовательность символов. В С++ строки описывают как массив элементов типа char. Например:

```
char s[25];  //Описана строка из 25 символов.
```

Структура - это тип данных, который позволяет объединить разнородные данные и обрабатывать их как единое целое.

Например

```
struct fraction //Объявлена структура правильная
дробь.
{
//Определяем поля структуры:
int num; //поле числитель,
int den; //поле знаменатель.
}
...
fraction d, D[20]; //Определена переменная d,
//массив D[20], типа fraction.
...
d.num;  //Обращение к полю num переменной d.
D[4].den;  //Обращение к полю den
//четвертого элемента массива D.
```

4.5 Указатели

Указатели широко применяются в С++. Можно сказать, что именно

наличие указателей сделало этот язык удобным для системного программирования. С другой стороны это одна из наиболее сложных для освоения возможностей С++. Идея работы с указателями состоит в том, что пользователь работает с адресом ячейки памяти.

Как правило, при обработке оператора объявления переменной

```
тип имя_переменной;
```

компилятор автоматически выделяет память под переменную имя_переменной в соответствии с указанным типом:

```
char C;//Выделена память под символьную переменную C
```

Доступ к объявленной переменной осуществляется по ее имени. При этом все обращения к переменной заменяются на адрес ячейки памяти, в которой хранится ее значение. При завершении программы или функции, в которой была описана переменная, память автоматически освобождается.

Доступ к значению переменной можно получить иным способом – определить собственные переменные для хранения адресов памяти. Такие переменные называют ***указателями***. С помощью указателей можно обрабатывать массивы, строки и структуры, создавать новые переменные в процессе выполнения программы, передавать адреса фактических параметров функциям и адреса функций в качестве параметров.

Итак, указатель это переменная, значением которой является адрес памяти, по которому хранится объект определенного типа (другая переменная). Например, если С это переменная типа char, а Р – указатель на С, значит в Р находится адрес по которому в памяти компьютера хранится значение переменной С.

Как и любая переменная, указатель должен быть объявлен. При объявлении указателей всегда указывается тип объекта, который будет храниться по данному адресу:

```
тип *имя_переменной;
```

Например:

```
int *p;  //По адресу, записанному в переменной p
//будет храниться переменная типа int
```

Звездочка в описании указателя, относится непосредственно к имени, поэтому чтобы объявить несколько указателей, ее ставят перед именем каждого из них:

```
float *x, y, *z;  //Описаны указатели
//на вещественное число – x и z,
//а так же вещественная переменная y.
```

5. ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ

Приступая к решению задач необходимо помнить, что: каждая переменная программы должна быть объявлена; объявления переменных обычно помещают в начале функции, сразу за заголовком. Следует обратить внимание, что хотя язык С++ допускает объявление переменных практически в любом месте

функции, объявлять переменные лучше всего в начале функции, снабжая инструкцию объявления кратким комментарием о назначении
переменной; инструкция объявления переменной выглядит так:

Тип ИмяПеременной;

Внструкцию объявления переменной можно использовать для инициализации переменной. В этом случае объявление переменной записывают следующим образом:

Тип ИмяПеременной = НачальноеЗначение;

Требования к созданию имени переменной:

- в имени переменной можно использовать буквы латинского алфавита и цифры (первым символом должна быть буква);
- компилятор C++ различает прописные и строчные буквы, поэтому, например, имена Summa и summa обозначают разные переменные;
- основными числовыми типами языка C++ являются int (целый) и float (дробный);
- после инструкции объявления переменной рекомендуется указывать назначение переменной.

6. ИНСТРУКЦИЯ ПРИСВАИВАНИЯ

Приступая к решению задач этого раздела необходимо помнить, что:

- инструкция присваивания предназначена для изменения значений переменных, в том числе и для вычислений «по формуле»;
- в отличие от большинства языков программирования, в C++ инструкция присваивания может быть записана несколькими способами, например вместо x = x + dx можно записать x+ = dx, а вместо i = i + 1 можно воспользоваться оператором инкремента и записать i++ ;
- значение выражения в левой части инструкции присваивания зависит от типа операндов и операции, выполняемой над операндами;
- результатом выполнения операции деления над целыми операндами является целое, которое получается отбрасыванием дробной части результата деления.

7. ВВОД И ВЫВОД ДАННЫХ

Ввод-вывод данных в языке C++ осуществляется либо с помощью функций ввода-вывода в стиле C, либо с использованием библиотеки классов C++. Преимущество объектов C++ в том, что они легче в использовании, особенно если ввод-вывод достаточно простой. Функции ввода-вывода унаследованные от C более громоздкие, но подходят для задач с форматированным выводом данных.

7.1 Функции ввода - вывода

Функция

```
printf(строка форматов, список выводимых переменных);
```

выполняет форматированный вывод переменных, указанных в списке, в соответствии со строкой форматов.

Функция

```
scanf(строка форматов, список адресов вводимых переменных);
```

выполняет ввод переменных, адреса которых указанных в списке, в соответствии со строкой форматов.

Строка форматов содержит символы, которые будут выводиться на экран или запрашиваться с клавиатуры и так называемые спецификации. *Спецификации* - это строки, которые начинаются символом % и выполняют управление форматированием:

```
% флаг ширина.точность модификатор тип
```

Параметры флаг, ширина, точность и модификатор в спецификациях могут отсутствовать. Значения параметров спецификаций приведены в таблице 2.10.

Таблица 2.10. Символы управления

Параметр	**Назначение**
Флаги	
-	Выравнивание числа влево. Правая сторона дополняется пробелами. По умолчанию выравнивание вправо.
+	Перед числом выводится знак «+» или «-»
Пробел	Перед положительным числом выводится пробел, перед отрицательным «–»
#	Выводится код системы счисления: 0 – перед восьмеричным числом, 0x (0X) перед шестнадцатеричным числом.
Ширина	
N	Ширина поля вывода. Если n позиций недостаточно, то поле вывода расширяется до минимально необходимого. Незаполненные позиции заполняются пробелами.
0n	То же, что и n, но незаполненные позиции заполняются нулями.
Точность	
Ничего	Точность по умолчанию
n	Для типов e, E, f выводить n знаков после десятичной точки
Модификатор	
h	Для d, i, o, u, x, X тип short int.
l	Для d, i, o, u, x, X тип long int.
Тип	
c	При вводе символьный тип char, при выводе один байт
d	Десятичное int со знаком.
i	Десятичное int со знаком
o	Восьмеричное int unsigned.

u	Десятичное int unsigned
x,X	Шестнадцатеричное int unsigned, при x используются символы a-f, при X – A - F.
f	Значение со знаком вида [-]dddd.dddd
e	Значение со знаком вида [-]d.dddde[+|-]ddd
E	Значение со знаком вида [-]d.ddddE[+|-]ddd.
g	Значение со знаком типа e или f в зависимости от значения и точности
G	Значение со знаком типа e или F в зависимости от значения и точности.
s	Строка символов.

Кроме того, строка форматов может содержать некоторые специальные символы,

которые приведены в таблице 2.11.

Таблица 2.11. Специальные символы

Символ	**Назначение**
\b	Сдвиг текущей позиции влево.
\n	Перевод строки.
\r	Перевод в начало строки, не переходя на новую строку.
\t	Горизонтальная табуляция.
\’	Символ одинарной кавычки.
\’’	Символ двойной кавычки.
\?	Символ ?

Первой строкой программы, в которой будут применяться функции ввода-вывода языка C, должна быть директива #include <stdio.h>. Заголовочный файл stdio.h содержит описание функций ввода-вывода.

Рассмотрим работу функций на примере следующей задачи.

Пример 2. Зная a, b, c – длины сторон треугольника, вычислить площадь S и периметр P этого треугольника.

Входные данные: a, b, c. Выходные данные: S, P.

Для вычисления площади применим формулу Герона:

$S = \sqrt{r \cdot (r-a) \cdot (r-b) \cdot (r-c)},\ r = \frac{a+b+c}{2}$.

Далее приведены две программы для решения данной задачи и результаты их работы (рис. 2.7 - 2.8). Сравните работу функций printf и scanf в этих программах.

Пример 2. Вариант первый

```
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std;
int main()
```

```
{
float a,b,c,S,r; //Описание переменных.
printf("a="); //Вывод на экран символов a=.
//В функции scanf для вычисления адреса
//переменной применяется операция &.
scanf("%f",&a); //Запись в переменную a значения
//введенного с клавиатуры.
printf("b="); //Вывод на экран символов b=.
scanf("%f",&b); //Запись в переменную b значения
//введенного с клавиатуры.
printf("c="); //Вывод на экран символов c=
scanf("%f",&c); //Запись в переменную c значения
//введенного с клавиатуры.
r=(a+b+c)/2; //Вычисление полупериметра.
S=sqrt(r*(r-a)*(r-b)*(r-c)); //Вычисление площади
//треугольника.
printf("S=%5.2f \t",S); //Вывод символов S=,
//значения S и символа
//табуляции \t.
//Спецификация %5.2f
//означает, что будет
//выведено вещественное
//число из пяти знаков,
//два из которых после точки.
printf("p=%5.2f \n",2*r); //Вывод символов p=,
//значения выражения 2*r
//и символа окончания строки.
//Оператор printf("S=%5.2f \t p=%5.2f \n",S,2*r);
//выдаст тот же результат.
return 0;
}
```

//Пример 2. Вариант второй

```
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std;
int main()
{
float a,b,c,S,r;
printf("Vvedite a, b, c \n"); //Вывод на экран
//строки символов.
scanf("%f%f%f",&a,&b,&c);  //Ввод значений.
r=(a+b+c)/2;
S=sqrt(r*(r-a)*(r-b)*(r-c));
```

```
printf("S=%5.2f \t p=%5.2f \n",S,2*r); //Вывод
//результатов.
return 0;
}
```

7.2 Объектно-ориентированные средства ввода-вывода.

Описание объектов для управления вводом-выводом содержится в заголовочном файле `iostream`. При подключении этого файла с помощью директивы `#include <iostream.h>` в программе автоматически создаются объекты-потоки `cin` для ввода с клавиатуры и `cout` для вывода на экран, а так же операции помещения в поток << и чтения из потока >>.

Итак, с помощью объекта `cin` и операции >> можно присвоить значение любой переменной. Например, если переменная i описана как целочисленная, то команда `cin>> i;` означает, что в переменную i будет записано некое целое число, введенное с клавиатуры. Если нужно ввести несколько переменных, следует написать `cin>>x>>y>>z;`.

Объект `cout` и операция << позволяют вывести на экран значение любой переменной или текст. Текст необходимо заключать в двойные кавычки, кроме того, допустимо применение специальных символов `\t` и `\n` (таблица 2.11). Запись

```
cout<<i;
```

означает вывод на экран значения переменной i. А команда

```
cout<<x<<"\t"<<y;
```

выведет на экран значения переменных x и y разделенные символом табуляции.

Пример 3. *Дано трехзначное число. Записать его цифры в обратном порядке и вывести на печать новое число.*

Разберем решение данной задачи на конкретном примере. Здесь будут использоваться операции целочисленной арифметики.

Пусть P=456. Вычисление остатка от деления числа P на 10 даст его последнюю цифру (количество единиц в числе P):

$$456 \% 10 = 6.$$

Операция деления нацело числа P на 10 позволит уменьшить количество разрядов и число станет двузначным:

$$456 / 10 = 45.$$

Остаток от деления полученного числа на 10 будет следующей цифрой числа P (количество десятков в числе P):

$$45 \% 10 = 5.$$

Последнюю цифру числа P (количество сотен) можно найти так:

$$456 / 100 = 4.$$

Так как в задаче требовалось записать цифры числа P в обратном порядке, значит в новом числе будет 6 сотен, 5 десятков и 4 единицы:

$$S = 6*100 + 5*10 + 4 = 654.$$

Далее приведен текст программы, реализующей данную задачу для любого трехзначного числа.

```
 #include <iostream.h>
int main(int argc, char *argv[])
{
unsigned int P, S; //Определение целочисленных
//переменных без знака.
cout<<"P=";  //Вывод на экран символов P=.
cin>>P; //Ввод заданного числа P.
S=P%10*100+P/10%10*10+P/100; //Вычисление нового числа S.
cout<<"S="<<S<<endl; //Вывод на экран символов S=
//и значения переменной S.
return 0;
}
```

8.ОПЕРАТОРЫ C++

Операторы используются для обработки данных. Операторы выполняют вычисления, сравнения, присваивание, а также множество других специфических задач. В C++ существует большое количество операторов. Они сведены в таблицу 1.

Оператор	Описание	Пример
Арифметические операторы		
+	Сложение	x=x+z;
-	Вычитание	x=y - z
*	Умножение	x=y * z
/	Деление	x=y / z
%	Остаток от деления	x=y%10
Операторы присваивания		
=	Присваивание	x=10;
+=	Сложение с присваиванием	x+=10; (то же, что и x=x+10;)
-=	Вычитание с присваиванием	x-=10;
*=	Умножение с присваиванием	x*=10;
/=	Деление с присваиванием	x/=10;
&=	Порязрядное И с присваиванием	x&=10;
|=	Поразрядное ИЛИ с присваиванием	x|=10;
Логические операторы		
&&	Логическое И	If (x && 0xFF) {…}
||	Логическое ИЛИ	If (x || 0xFF) {…}
Операторы отношения		
==	Равно	If (x == 0xFF) {…}
!=	Не равно	If (x != 0xFF) {…}
<	Меньше	If (x < 0xFF) {…}

>	Больше	If (x > 0xFF) {…}
<=	Меньше или равно	If (x <= 0xFF) {…}
>=	Больше или равно	If (x >= 0xFF) {…}
Унарные операторы		
*	Косвенная адресация	int x=*y;
?	Взятие адреса	int *x=&y;
-	Поразрядное НЕ	x &=-0x02;
!	Логическое НЕ	if (!valid) {…}
++	Инкремент	x++; (то же, что и x=x+1;)
--	Декремент	x--;
Операторы классов и структур		
::	Разрешение области видимости	MyClass::SomeFunction();
->	Косвенный доступ	MyClass->SomeFunction();
.	Прямой доступ	MyClass.SomeFunction();

Пример 4. ***Деление простых дробей***

$$p = \frac{a}{в} : \frac{c}{d} = \frac{a \cdot d}{b \cdot c} = \frac{m}{n}.$$

```
void main()
{  int a, b, c, d;
  float p, m, n;
  cout<<"\na=";
  cin>>a;
  cout<<"\nb=";
  cin>>b;
  cout<<"\nc=";
  cin>>c;
  cout<<"\nd=";
  cin>>d;
         m=a*d;
   n=b*c;
  cout<<"\nm="<<m;
  cout<<"\nn="<<n;
  p=m/n;
  cout<<"\np="<<p<<endl;}
```

Пример 5. Операции увеличения и уменьшения на 1 (++ и --)

5.1 Префиксное увеличение на 1

```
#include <iostream.h>
void main()
{  int x=3,y=4,c;
      c=++x*++y;
  cout<<"\nc="<<c;
cout<<"\nx="<<x;
cout<<"\ny="<<y<<endl;}
```

5.2 Постфиксное увеличение на 1

```
#include <iostream.h>
void main()
{  int x=3,y=4,c;
      c=x++*y++;
  cout<<"\nc="<<c;
cout<<"\nx="<<x;
cout<<"\ny="<<y<<endl;}
```

4.3 Префиксное уменьшение на 1

```
#include <iostream.h>
void main()
{  int x=3,y=4,c;
      c=--x*--y;
  cout<<"\nc="<<c;
cout<<"\nx="<<x;
cout<<"\ny="<<y<<endl;}
```

5.4 Постфиксное уменьшение на 1

```
#include <iostream.h>
void main()
{  int x=3,y=4,c;
      c=x--*y--;
  cout<<"\nc="<<c;
cout<<"\nx="<<x;
cout<<"\ny="<<y<<endl;}
```

9. БИБЛИОТЕКА МАТЕМАТИЧЕСКИХ ФУНКЦИЙ

При вычислении стандартных математических функций, перечисленных в таблице 2, необходимо к программе подключить библиотеку математических функций. Для подключения этой библиотеки используется директива препроцессора `#include <math.h>`. Здесь `math.h` – имя заголовочного файла этой библиотеки.

Обращение	**Функция**	**Тип аргумента**	**Тип результата**
abs(x)	Абсолютное значение целого числа	int	int
acos(x)	Арккосинус(радианы)	double	double
asin(x)	Арксинус(радианы)	double	double
atan(x)	Арктангенс(радианы)	double	double
ceil(x)	Округляет до ближайшего целого, не меньшего x (округляет вверх)	double	double
cos(x)	Косинус (x в радианах)	double	double
cosh(x)	Гиперболический косинус	double	double
exp(x)	Экспонента от x	double	double
fabs(x)	Абсолютное значение вещественного x	double	double
floor(x)	Округляет до ближайшего целого, не превышающего x (округляет вниз)	double	double
fmod(x,y)	Остаток от деления нацело x на y	double	double
frexp(double x, int expptr)	Выделяет из числа мантиссу и целую часть	double	double
ldexp	Преобразует мантиссу и показатель степени в число (противоположна функции frexp)	double	double
log(x)	Логарифм натуральный (lnx)	double	double
log10(x)	Логарифм десятичный (logx)	double	double
modf(double x, double intprt)	Разбивает число на целую и дробную части	double	double
pow(x,y)	Возводит x в степень y	double	double
sin(x)	Синус (x в радианах)	double	double
sinh(x)	Гиперболический синус	double	double
sqrt(x)	Корень квадратный (положительное значение)	double	double

tan(x)	Тангенс (x в радианах)	double	double
tanh(x)	Гиперболический тангенс	double	double

Пример 6. Написать консольную программу, вычисляющую значение функции

$$y=\frac{x^2-3\sqrt{x+1}+\ln 2x^2}{3+\sin\frac{2x^2+1}{2+\sqrt{x}}}$$

Код программы:

```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
#pragma hdrstop

//---------------------------------------------------------------------------

#pragma argsused
int main(int argc, char* argv[])
{
        double x=2,y;
        y=(pow(x,2)-3*sqrt(x+1)+log(2*pow(x,2)))/(3+sin((2*pow(x,2)+1)/(2+sqrt(x))));
        cout    <<"y="<<y;

        getch();

}
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1 Ввод-вывод данных. Операция присваивания.

Разработать программу на языке C++. Все входные и выходные данные в задачах — вещественные числа. Для ввода и вывода данных использовать функции scanf и printf.

1. Даны катеты прямоугольного треугольника a и b. Найти гипотенузу c и углы треугольника α, β, χ.
2. Известна гипотенуза c и прилежащий угол α прямоугольного треугольника. Найти площадь треугольника S и угол β.
3. Известна диагональ квадрата d. Вычислить площадь S и периметр P квадрата.
4. Дан диаметр окружности d. Найти ее длину L и площадь круга S.
5. Даны три числа – a, b, c. Найти их среднее арифметическое и среднее геометрическое.
6. Даны катеты прямоугольного треугольника a и b. Найти его гипотенузу c и периметр P.

7. Дан длина окружности L. Найти ее радиус R и площадь круга S.
8. Даны два ненулевых числа a и b. Найти сумму S, разность R, произведение P и частное d их квадратов.
9. Поменять местами содержимое переменных A и B и вывести новые значения A и B.
10. Точки A и B заданы координатами на плоскости: A(x1,y1), B(x2,y2). Найти длину отрезка AB.
11. Заданы два катета прямоугольного треугольника a и b. Вычислить его площадь S и периметр P.
12. Даны переменные A, B, C. Изменить их значения, переместив содержимое A в B, B — в C, C — в A, и вывести новые значения переменных A, B, C.
13. Известна диагональ ромба d. Вычислить его площадь S и периметр P.
14. Найти значение функции y=4·(x+1)3+5·(x−1)5+2 и ее производной при заданном значении x.
15. Даны два ненулевых числа a и b. Найти сумму S, разность R, произведение P и частное D их модулей.
16. Известны координаты вершин квадрата ABCD: A(x1,y1) и C(x2,y2). Найти его площадь S и периметр P.
17. Даны длины сторон прямоугольника a и b. Найти его площадь S и периметр P.
18. Известно значение периметра P равностороннего треугольника. Вычислить его площадь S.
19. Задан периметр квадрата P. Вычислить сторону квадрата a, диагональ d и площадь S.
20. Дана сторона квадрата a. Вычислить периметр квадрата P, его площадь S и длину диагонали d.
21. Три точки заданы координатами на плоскости: A(x1,y1), B(x2,y2) и C(x3,y3). Найти длины отрезков AB и BC.
22. Даны переменные A, B, C. Изменить их значения, переместив содержимое A в C, C — в B, B — в A, и вывести новые значения переменных A, B, C.
23. Даны числа – a1, a2, a3, a4, a5. Найти их среднее арифметическое и среднее геометрическое значения.
24. Найти значение функции $y = \frac{3}{2}(x+3)^4 - \frac{1}{5}(x-1)^5$ и ее производной при заданном значении x.
25. Точки A и B заданы координатами в пространстве: A(x1,y1, z1), B(x2,y2, z2). Найти длину отрезка AB.

2 Операции целочисленной арифметики.

Разработать программу на языке C++. Все входные данные в задачах — целые числа. Для ввода и вывода данных использовать объектно-ориентированные средства ввода-вывода.
1. Расстояние L задано в сантиметрах. Найти количество полных метров в нем и остаток в сантиметрах.

2. Масса M задана в килограммах. Найти количество полных тонн в ней и остаток в килограммах.
3. Дан размер файла B в байтах. Найти количество полных килобайтов, которые занимает данный файл и остаток в байтах.
4. Дано двузначное число. Вывести на печать количество десятков и единиц в нем.
5. Дано двузначное число. Найти сумму его цифр.
6. Дано двузначное число. Найти произведение его цифр.
7. Дано двузначное число. Вывести число, полученное при перестановке цифр исходного числа.
8. Дано трехзначное число. Определить сколько в нем единиц, десятков и сотен.
9. Дано трехзначное число. Найти сумму его цифр.
10. Дано трехзначное число. Найти произведение его цифр.
11. Дано трехзначное число. Вывести число, полученное при перестановке цифр сотен и десятков исходного числа.
12. Дано трехзначное число. Вывести число, полученное при перестановке цифр сотен и единиц исходного числа.
13. Дано трехзначное число. Вывести число, полученное при перестановке цифр десятков и единиц исходного числа.
14. С начала суток прошло N секунд. Найти количество полных минут, прошедших с начала суток и остаток в секундах.
15. С начала суток прошло N секунд. Найти количество полных часов, прошедших с начала суток и остаток в секундах.
16. Дано двузначное число ($a \leqslant 88$). Вывести на печать число, которое получится если каждую цифру числа a увеличить на единицу.
17. Дано двузначное число ($a \geq 22$). Вывести на печать число, которое получится если каждую цифру числа a уменьшить на единицу.
18. Расстояние L задано в метрах. Найти количество полных километров в нем и остаток в метрах.
19. Масса M задана в граммах. Найти количество полных килограммов в ней и остаток в граммах.
20. Размер файла B дан в килобайтах. Найти количество полных мегабайтов, которые занимает данный файл и остаток в килобайтах.
21. Расстояние L задано в дециметрах. Найти количество полных метров в нем и остаток в сантиметрах.
22. С начала года прошло K дней. Найти количество полных недель, прошедших с начала года и остаток в днях.
23. С начала года прошло K часов. Найти количество полных дней, прошедших с начала года и остаток в часах.
24. Дано двузначное число ($a \leqslant 44$). Вывести на печать число, которое получится если удвоить каждую цифру числа a.
25. Дано двузначное число ($a \geq 22$). Вывести на печать число, которое получится если каждую цифру числа a уменьшить вдвое.

3 Встроенные математические функции

Создать консольное приложение для расчета значений переменных y и z по заданным формулам. В программе предусмотреть ввод исходных данных с клавиатуры. (Предварительно вычислите ожидаемые значения y и z с помощью калькулятора. Убедитесь, что значения, вычисленные с помощью калькулятора, совпадают с результатами, которые получаются в результате работы программы).

Вариант 1 $y=\frac{\sin\left(\frac{\pi}{2}+3\alpha\right)}{1-\sin(3\alpha-\pi)}\ ;\ z=ctg\left(\frac{5}{4}\pi+\frac{3}{2}\alpha\right)$	**Вариант 2** $y=\cos\alpha+\sin\alpha+\cos 3\alpha+\sin 3\alpha$ $z=2\sqrt{2}\cos\alpha\cdot\sin\left(\frac{\pi}{4}+2\alpha\right)$
Вариант 3 $y=\frac{\sin 2\alpha+\sin 5\alpha-\sin 3\alpha}{\cos\alpha+1-2\sin^2 2\alpha}$ $z=2\sin\alpha$	**Вариант 4** $y=\frac{\sin 2\alpha+\sin 5\alpha-\sin 3\alpha}{\cos\alpha-\cos 3\alpha+\cos 5\alpha}$ $z=tg\,3\alpha$
Вариант 5 $y=1-\frac{1}{4}\sin^2 2\alpha+\cos 2\alpha$ $z=\cos^2\alpha+\cos^4\alpha$	**Вариант 6** $y=\cos\alpha+\cos 2\alpha+\cos 6\alpha+\cos 7\alpha$ $z=4\cos\frac{\alpha}{2}\cdot\cos\frac{5}{2}\alpha\cdot\cos 4\alpha$
Вариант 7 $y=\cos^2\left(\frac{3}{8}\pi-\frac{\alpha}{4}\right)-\cos^2\left(\frac{11}{8}\pi+\frac{\alpha}{4}\right)$ $z=\frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$	**Вариант 8** $y=2\cdot\sin^2(3\pi-2\alpha)\cos^2(5\pi+2\alpha)$ $z=\frac{1}{4}-\frac{1}{4}\sin\left(\frac{5}{2}\pi-8\alpha\right)$

Вариант 9

$$y=(\cos\alpha-\cos\beta)^2-(\sin\alpha-\sin\beta)^2$$

$$z=-4\cdot\sin^2\frac{\alpha-\beta}{2}\cdot\cos(\alpha+\beta)$$

Вариант 10

$$y=\cos^4 a+\sin^2 b+\frac{1}{4}\sin^2 2a-1$$

$$z=\sin(b+a)\cdot\sin(b-a)$$

Вариант 11

$$y=\frac{1-2\sin^2\alpha}{1+\sin 3\alpha}$$

$$z=\frac{1-tg\,\alpha}{1+tg\,\alpha}$$

Вариант 12

$$y=\frac{\sin 4\alpha}{1+\cos 4\alpha}\cdot\frac{\cos 2\alpha}{1+\cos 2\alpha}$$

$$z=ctg\left(\frac{3}{2}\pi-\alpha\right)$$

Вариант 13

$$y=\frac{\sin\alpha+\cos(2\beta-\alpha)}{\cos\alpha-\sin(2\beta-\alpha)}$$

$$z=\frac{1+\sin 2\beta}{\cos 2\beta}$$

Вариант 14

$$y=\frac{1}{4}\cdot(\sin(\alpha+\beta-\gamma)-\sin(\beta+\gamma-\alpha)+$$

$$+\sin(\gamma+\alpha-\beta)-\sin(\alpha+\beta+\gamma))$$

$$z=\sin\alpha\cdot\cos\beta\cdot\cos\gamma$$

Вариант 15

$$y=\frac{\sqrt{2b+2\sqrt{b^2-4}}}{\sqrt{b^2-4}+b+2}$$

$$z=\frac{1}{\sqrt{b+2}}$$

Вариант 16

$$y=\frac{x^2+2x-3+(x+1)\cdot\sqrt{x^2-9}}{x^2+2x-3+(x-1)\cdot\sqrt{x^2-9}}$$

$$z=\sqrt{\frac{x+3}{x-3}}$$

Вариант 17

$$y=\frac{1}{4}\cdot[\cos(\alpha+\beta-\gamma)+\cos(\beta+\gamma-\alpha)+$$

$$+\cos(\gamma+\alpha-\beta)+\cos(\alpha+\beta+\gamma)]$$

$$z=\cos\alpha\cdot\cos\beta\cdot\cos\gamma$$

Вариант 18

$$y=\frac{3\ tg\,\alpha-tg^3\alpha}{1-3\ tg^2\alpha}$$

$$z=tg\,3\alpha$$

Вариант 19

$$y=\sqrt{\frac{1-\cos\alpha}{1+\cos\alpha}}$$

$$z=\frac{1-\cos\alpha}{\sin\alpha}$$

Вариант 20

$$y=\left(\frac{1+a+a^2}{2a+a^2}+2-\frac{1-a+a^2}{2a-a^2}\right)^{-1}\cdot(5-2a^2)$$

$$z=\frac{4-a}{2}$$

Вариант 21

$$y=\frac{1}{8}(\cos 4\alpha+4\cos 2\alpha+3)$$

$$z=\cos^4\alpha$$

Вариант 22

$$y=\frac{4\ tg\,\alpha-4\ tg^3\alpha}{1-6\ tg^2\alpha+tg^4\alpha}$$

$$z=tg\,4\alpha$$

Вариант 23

$$y=\frac{1}{8}(\cos 4\alpha-4\cos 2\alpha+3)$$

$$z=\sin^4\alpha$$

Вариант 24

$$y=\frac{\cos\alpha+\sin\alpha}{\cos\alpha-\sin\alpha}$$

$$z=tg\,2\alpha+\sec 2\alpha$$

Вариант 25

$$y=\frac{1}{4}\cdot[\sin(\alpha+\beta-\gamma)+\sin(\beta+\gamma-\alpha)+$$

$$+\sin(\gamma+\alpha-\beta)-\sin(\alpha+\beta+\gamma)]$$

$$z=\sin\alpha\cdot\sin\beta\cdot\sin\gamma$$

Вариант 26

$$y=\frac{1}{4}(3\cdot\sin\alpha-\sin 3\alpha)$$

$$z=\sin^3\alpha$$

ТЕМА 3. ПРОГРАММЫ НА ЯЗЫКЕ С++ СО СТРУКТУРОЙ ВЕТВЛЕНИЯ

1. ЦЕЛЬ ЗАНЯТИЯ

Целью ЗАНЯТИЯ является приобретение знаний и навыков составления программ с использованием операторов ветвления if и switch и операторов цикла for, do while и while.

2. ОПЕРАТОРЫ ВЕТВЛЕНИЯ

2.1. Условный оператор if

Условный оператор if используется для разветвления процесса вычислений на два направления. Формат оператора:

```
if ( выражение ) оператор _1; [else оператор _2;]
```

Сначала вычисляется выражение, которое может иметь арифметический тип или тип указателя. Если оно не равно нулю (имеет значение `true`), выполняется первый оператор, иначе второй. После этого управление передается на оператор, следующий за условным (рис. 1, а). Одна из ветвей может отсутствовать, логичнее опускать вторую ветвь вместе с ключевым словом `else` (рис. 1, б). Если в какой-либо ветви требуется выполнить несколько операторов, их необходимо заключить в блок, иначе компилятор не сможет понять, где заканчивается ветвление. Блок может содержать любые операторы, в

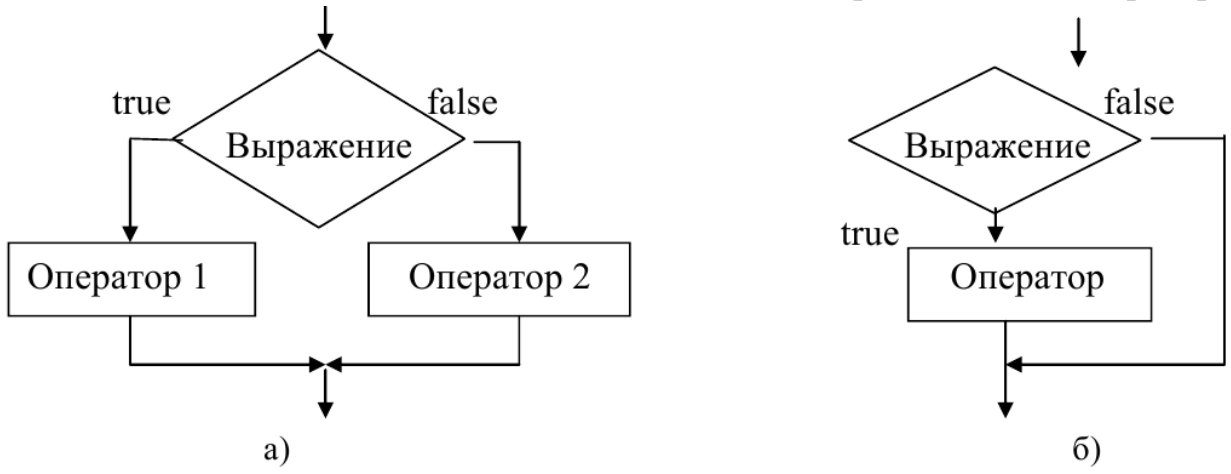


Рис. 1. Структурная схема условного оператора

том числе описания и другие условные операторы, (но не может состоять из одних описаний). Необходимо указывать, что переменная, описанная в блоке, вне блока не существует.

Итак, приступая к решению задач данного раздела, следует помнить, что:

- оператор `if` используется для выбора одного из двух направлений дальнейшего хода программы;

- выбор последовательности выполнения осуществляется в зависимости от значения условия – заключенного в скобки выражения, записанного после `if`;
- инструкция, записанная после `else`, выполняется в том случае, если значение выражения условие равно нулю, во всех остальных случаях выполняется инструкция, следующая за условием;
- если при соблюдении или несоблюдении условия надо выполнить несколько инструкций программы, то эти инструкции следует объединить в группу – заключить в фигурные скобки;
- при помощи вложенных одна в другую нескольких инструкций `if` можно реализовать множественный выбор.

Пример 1. *Присвоить переменной b единицу, если a<b. В противном случае значения a и b остаются неизменными.*

В примере 1 отсутствует ветвь `else` (рис. 2). Подобная конструкция

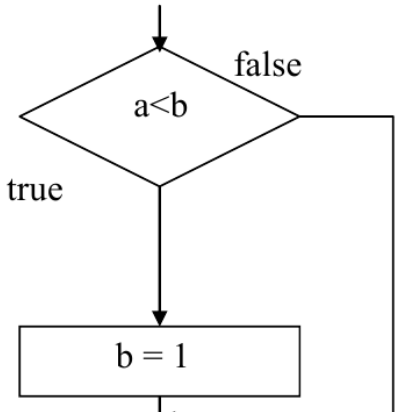


Рис. 2. Структурная схема алгоритма к примеру 1

называется «пропуск оператора», поскольку присваивание либо выполняется, либо пропускается в зависимости от выполнения условия.

```
#include <iostream.h>
void main()
{  int a,b;
  cout<<"a=";cin>>a;
  cout<<"b="; cin>>b;
    if (a<b) b=1;
  cout<<"a="<<a<<"\n";
  cout<<"b="<<b<<"\n";}
```

Пример 2. *Найти наименьшее значение из трех переменных a, b, c*

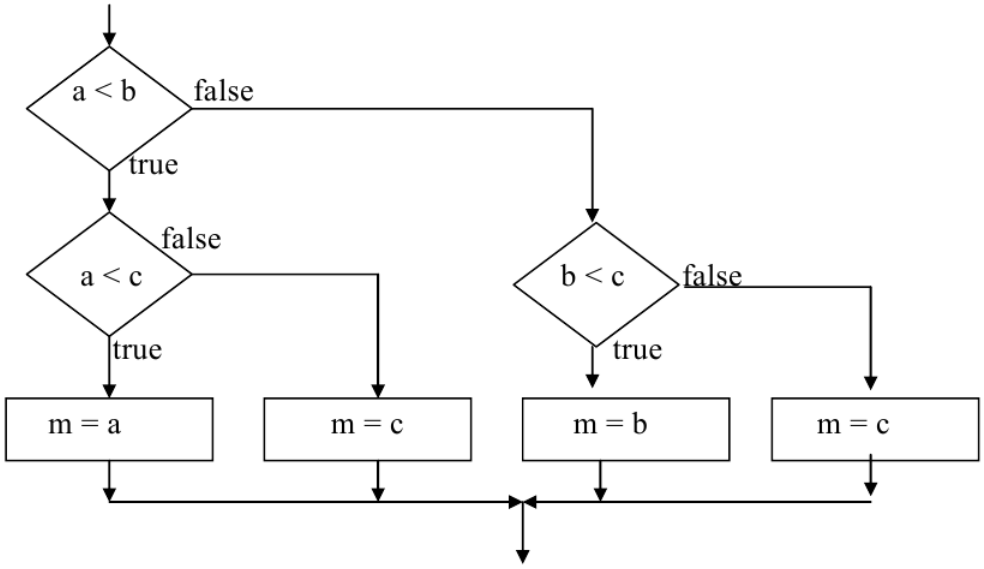


Рис. 3. Структурная схема алгоритма к примеру 2

Оператор в примере 2 вычисляет наименьшее значение из трех переменных. Фигурные скобки в данном случае не обязательны, т.к. компилятор относит часть `else` к ближайшему `if`.

```
#include <iostream.h>
void main()
{  int a, b, c, m;
cout<<"a=";cin>>a;
cout<<"b=";cin>>b;
cout<<"c=";cin>>c;
    if (a<b)
{  if (a<c) m=a; else m=c;} else {if (b<c) m=b; else
m=c;}
cout<<"m="<<m<<endl;}
```

Пример 3. *Дано вещественное число x. Для функции, график которой приведен на рис. 4 вычислить y=f(x).*

Аналитически функцию представленную на рис. 4 можно записать так:

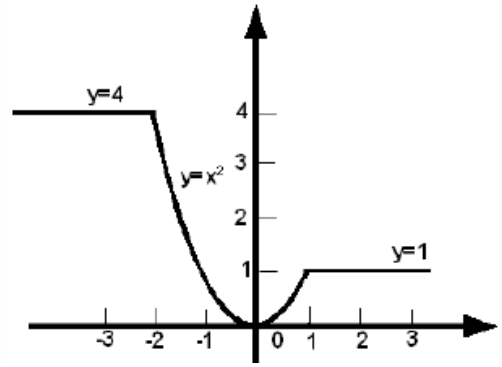
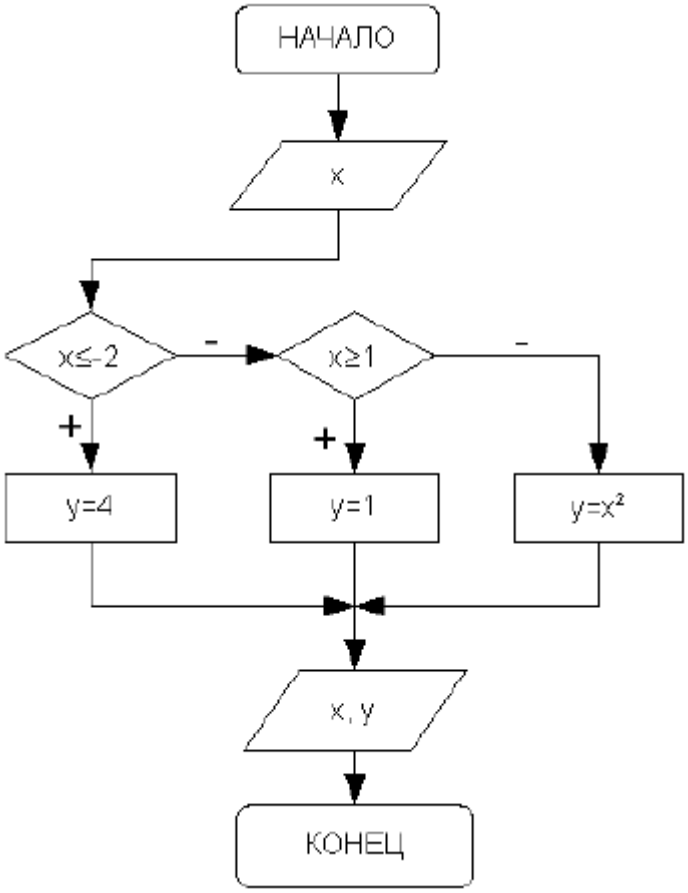


Рис. 4. Графическое представление примера 3 .

$$y(x) = \begin{cases} 4, & \text{если } x \le -2 \\ 1, & \text{если } x \ge 1 \\ x^2, & \text{если } -2 < x < 1 \end{cases}$$




```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop

//---------------------------

#pragma argsused
int main(int argc, char* argv[])
{
float X,Y;
cout<<"X="; cin>>X;
if (X<=-2) Y=4;
else if (X>=1) Y=1;
else Y=X*X;
cout <<"Y=" <<Y<< endl;
getch();
}
```

Рис. 5. Блок-схема алгоритма решения примера 3

Пример 4. *Даны вещественные числа x и y. Определить принадлежит ли точка с координатами (x; y) заштрихованной части плоскости (рис. 6).*

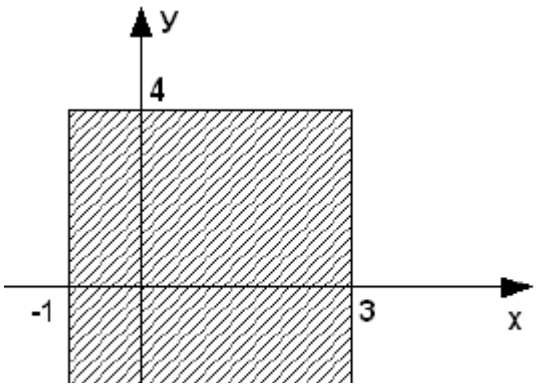


Рис. 6. Графическое представление примера 4

Как показано на рис. 6 плоскость ограничена линиями x=-1, x=3, y=-2 и y=4. Значит точка с координатами (x; y) будет принадлежать этой плоскости, если будут выполняться следующие условия: $x \geq -1$, $x \leq 3$, $y \geq -2$ и $y \leq 4$. Иначе точка лежит за пределами плоскости.

Блок-схема, описывающая алгоритм решения задачи представлена на рис. 7.

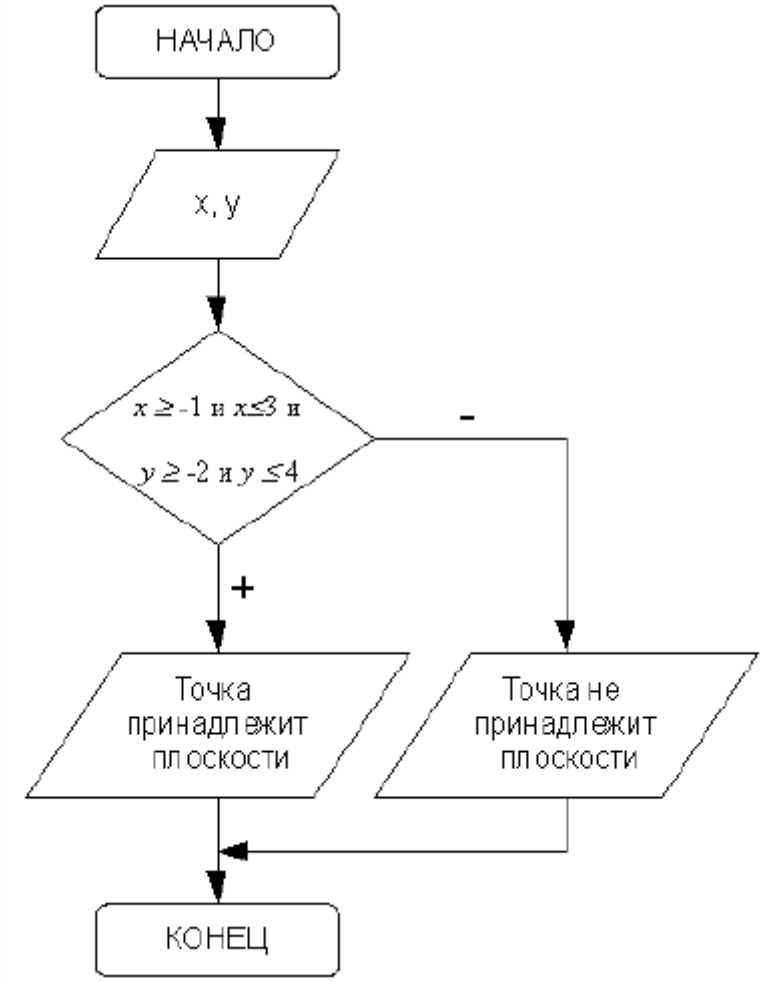


Рис. 7. Блок схема решения примера 4

```
#include <iostream.h>
int main()
{float X,Y;
cout<<"X="; cin>>X;
cout<<"Y="; cin>>Y;
 if (X>=-1 && X<=3 && Y>=-2 && Y<=4)
cout << "Точка лежит в заданной области" << endl;
else
cout<<"Точка лежит вне заданной области "<<endl;
return 0;
}
```

Пример 5. *Даны вещественные числа x и y. Определить, принадлежит ли точка с координатами (x; y) заштрихованной части плоскости (рис. 8).*

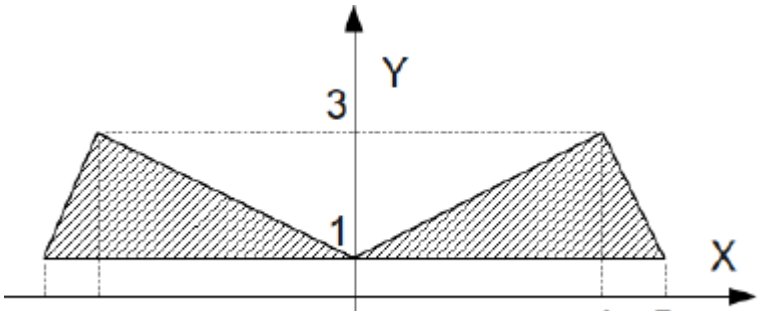


Рис. 8. Графическое представление примера 5

Составим уравнения линий, ограничивающих заданные плоскости. В общем виде уравнение прямой проходящей через точки с координатами (x_1, y_1) и (x_2, y_2) имеет вид:

$$\frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1}$$

Треугольник в первой координатной области ограничен линиями, проходящими
через точки:
1. (0, 1) — (4, 3);
2. (4, 3) — (5, 1);
3. (5, 1) — (0, 1).

Следовательно, уравнение первой линии

$$\frac{x-0}{4-0} = \frac{y-1}{3-1} \Rightarrow \frac{x}{4} = \frac{y-1}{2} \Rightarrow y = 1 + \frac{1}{2} \cdot x$$

Уравнение второй линии:

$$\frac{x-4}{5-4} = \frac{y-3}{1-3} \Rightarrow x - 4 = \frac{y-3}{-2} \Rightarrow 2x + 8 = y - 3 \Rightarrow y = -2 \cdot x + 11$$

Уравнение третьей линии y=1.

Линии, которые формируют треугольник во второй координатной области, проходят через точки:
4. (0, 1) — (-4, 3);
5. (-4, 3) — (-5, 1);
6. (-5, 1) — (0, 1);

Следовательно, уравнение первой линии

$$\frac{x-0}{-4-0} = \frac{y-1}{3-1} \Rightarrow \frac{x}{-4} = \frac{y-1}{2} \Rightarrow y = 1 - \frac{1}{2} \cdot x$$

Уравнение второй линии:

$\frac{x+4}{-5+4} = \frac{y-3}{1-3} \Rightarrow \frac{x+4}{-1} = \frac{y-3}{-2} \Rightarrow -2x - 8 = -y + 3 \Rightarrow y = 2 \cdot x + 11$

Уравнение третьей линии y=1.

Таким образом, условие попадания точки в заштрихованную часть плоскости имеет вид:

$$\begin{cases} y \le 1 + \frac{1}{2} \cdot x \\ y \le -2 \cdot x + 11 \\ y \ge 1 \end{cases} \text{ или } \begin{cases} y \le 1 - \frac{1}{2} \cdot x \\ y \le 2 \cdot x + 11 \\ y \ge 1 \end{cases}$$

Текст программы:

```
#include <iostream>
int main()
{
float X,Y;
cout<<"X="; cin>>X;
cout<<"Y="; cin>>Y;
if ((Y<=1+(float)1/2*X && Y<=-2*X+11 && Y>=1) ||
(Y<=1-(float)1/2*X && Y<=2*X+11 && Y>=1))
cout << " Точка лежит в заданной области " << endl;
else
cout<<" Точка лежит вне заданной области "<<endl;
return 0;
}
```

Пример 6. *Написать программу решения квадратного уравнения* $ax^2+bx+c=0$.
Результаты работы программы: вещественные числа x1 и x2 – корни квадратного уравнения либо сообщение о том, что корней нет.

Вспомогательные переменные: вещественная переменная d, в которой будет храниться дискриминант квадратного уравнения.
Составим словесный алгоритм решения этой задачи.
1. Начало алгоритма
2. Ввод числовых значений переменных a, b и c.
3. Вычисление значения дискриминанта d по формуле $d = b^2 - 4ac$.
4. Если d<0, то переход к п.5, иначе переход к п.6.
5. Вывод сообщения Корней нет и переход к п.8.
6. Вычисление корней $x1 = \frac{-b+\sqrt{d}}{2a}$ и $x2 = \frac{-b-\sqrt{d}}{2a}$
7. Вывод значений x1 и x2 на экран
8. Конец алгоритма.

Блок-схема, соответствующая этому описанию представлена на рис. 9.

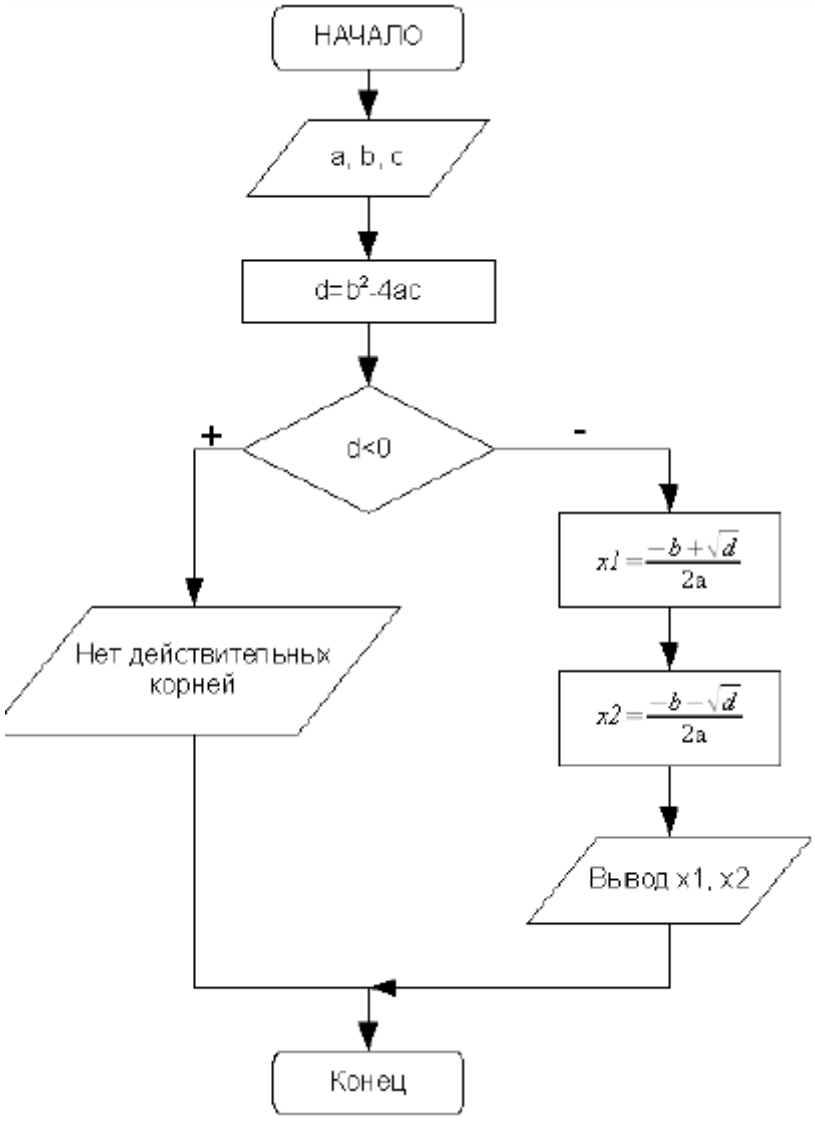


Рис. 9. Алгоритм решения квадратного уравнения

```
#include <iostream>
#include <math.h>
int main()
{
float a,b,c,d,x1,x2;
//Ввод значений коэффициентов
//квадратного уравнения.
cout<<"a=";cin>>a;
cout<<"b=";cin>>b;
cout<<"c=";cin>>c;
d=b*b-4*a*c; //Вычисление дискриминанта.
if (d<0) //Если дискриминант отрицательный,
//то вывод сообщения, о том
//что корней нет,
```

```
cout<<"Действительные корни отсутствуют";
 else
{
//иначе вычисление корней
x1=(-b+sqrt(d))/2/a;
x2=(-b-sqrt(d))/(2*a);
//и вывод их значений.
cout<<"X1="<<x1<<"\t X2="<<x2<<"\n";
}
return 0;
}
```

Пример 7. *Составить программу нахождения действительных и комплексных корней квадратного уравнения* $ax^2+bx+c=0$.

Исходные данные: вещественные числа a, b и c – коэффициенты квадратного уравнения.

Результаты работы программы: вещественные числа x1 и x2 – действительные корни квадратного уравнения либо x1 и x2 – действительная и мнимая части комплексного числа.

Вспомогательные переменные: вещественная переменная d, в которой будет храниться дискриминант квадратного уравнения.

Можно выделить следующие этапы решения задачи:

1. Ввод коэффициентов квадратного уравнения a, b и c.
2. Вычисление дискриминанта d по формуле $d=b^2-4ac$.
3. Проверка знака дискриминанта. Если $d\geq 0$, то вычисление действительных корней:

$$x1=\frac{-b+\sqrt{d}}{2a} \text{ и } x2=\frac{-b-\sqrt{d}}{2a}$$

и вывод их на экран. При отрицательном дискриминанте выводится сообщение о том,

что действительных корней нет, и вычисляются комплексные корни (*Комплексные числа записываются в виде a+ib, где a – действительная часть комплексного числа, b –*

мнимая часть комплексного числа, i – мнимая единица $\sqrt{-1}$).

$$x1=-\frac{b}{2a}+i\frac{\sqrt{|d|}}{2a} \text{ и } x2=-\frac{b}{2a}-i\frac{\sqrt{|d|}}{2a}$$

У обоих комплексных корней действительные части одинаковые, а мнимые отличаются знаком. Поэтому можно в переменной x1 хранить действительную

часть числа $-\frac{b}{2a}$, в переменной x2 – модуль мнимой части $\frac{\sqrt{|d|}}{2a}$, а в качестве корней вывести

x1 + i x2 и x1 — i x2.

На рис. 10 изображена блок-схема решения задачи. Блок 1 предназначен для

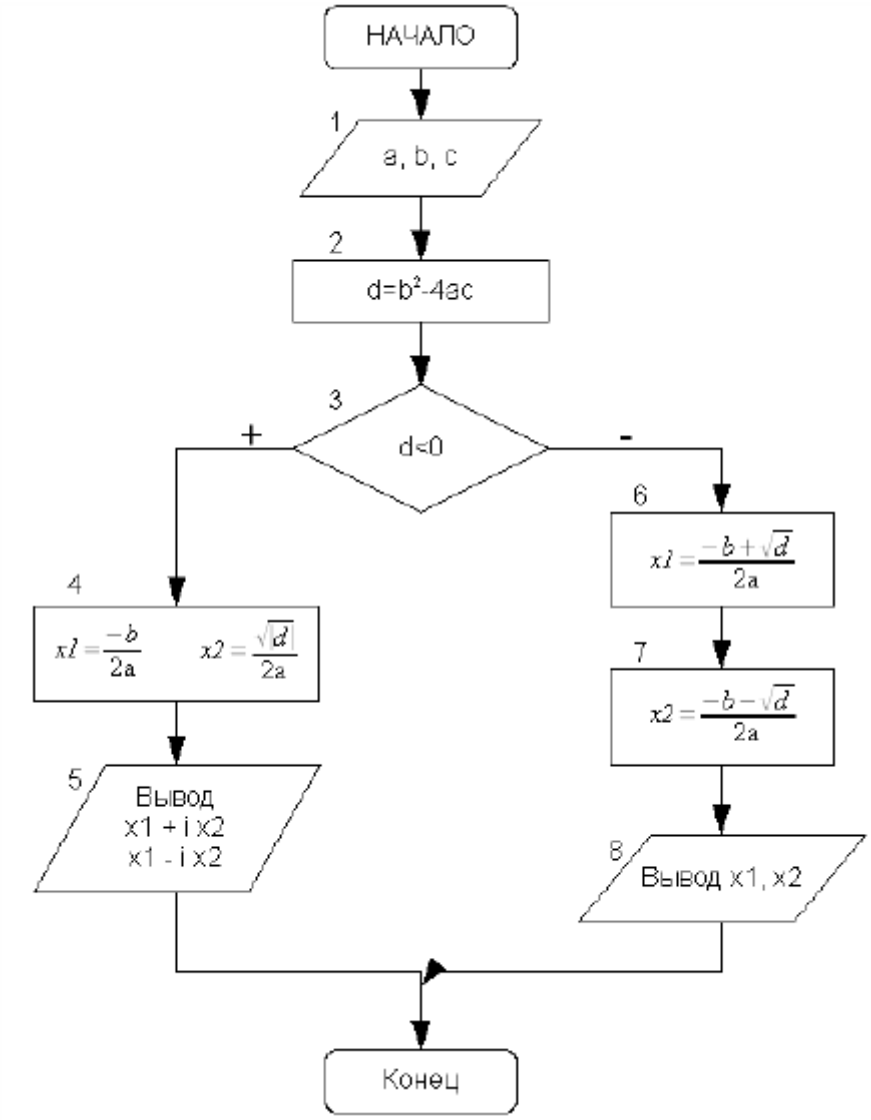


Рис. 10. Алгоритм нахождения действительных или комплексных корней квадратного уравнения

ввода коэффициентов квадратного уравнения. В блоке 2 осуществляется вычисление дискриминанта. Блок 3 осуществляет проверку знака дискриминанта, если дискриминант отрицателен, то корни комплексные, их расчет происходит в блоке 4 (действительная часть корня записывается в переменную x1, модуль мнимой – в переменную x2), а вывод – в блоке 5 (первый корень x1 + i x2, второй – x1 — i x2). Если дискриминант положителен, то вычисляются действительные корни уравнения (блок 6) и выводятся на экран (блок 7).

Текст программы, реализующей поставленную задачу:

```
#include <iostream>
#include <math.h>
int main()
{ float a,b,c,d,x1,x2;
cout<<"a=";cin>>a;
cout<<"b=";cin>>b;
cout<<"c=";cin>>c;
d=b*b-4*a*c;
if (d<0)
     { //Если дискриминант отрицательный,
     //то вывод соответствующего сообщения.
     cout<<"Real roots are not present \n";
     x1=-b/(2*a); //Вычисление действительной
     //части комплексных корней.
     x2=sqrt(fabs(d))/(2*a); //Вычисление модуля
                             //мнимой части
                             //комплексных корней
     //Сообщение о комплексных корнях
     //уравнения вида ax^2+bx+c=0.
     cout<<"Complex roots of equalization \n";
     cout<<a<<"x^2+"<<b<<"x+"<<c<<"=0 \n";

     //Вывод значений комплексных
     //корней в виде x1±ix2
     cout<<x1<<"+i*("<<x2<<")\t";
     cout<<x1<<"-i*("<<x2<<")\n";
}
else
{
     //Если дискриминант положительный,
     //вычисление действительных корней
     //и вывод их на экран.
     x1=(-b+sqrt(d))/2/a;
     x2=(-b-sqrt(d))/(2*a);
     cout<<"Real roots of equalization \n";
     cout<<a<<"x^2+"<<b<<"x+"<<c<<"=0 \n";
     cout<<"X1="<<x1<<"\t X2="<<x2<<"\n";
}
return 0;
}
```

Пример 8. *Составить программу для решения кубического уравнения* $ax^3+bx^2+cx+d=0$.

Кубическое уравнение имеет вид:

$$ax^3+bx^2+cx+d=0 \qquad (3.1)$$

После деления на a уравнение (3.1) принимает канонический вид:

$$x^3+rx^2+sx+t=0 \qquad (3.2)$$

где $r=\frac{b}{a}$, $s=\frac{c}{a}$, $t=\frac{d}{a}$.

В уравнении (3.2) сделаем замену $x=y-\frac{r}{3}$, и получим приведенное уравнение:

$$y^3+py+q=0 \qquad (3.3)$$

где $p=\frac{3s-r^2}{3}$, $q=\frac{2r^3}{27}-\frac{rs}{3}+t$.

Число действительных корней приведенного уравнения (3.3) зависит от знака дискриминанта (табл. 3.1):

$$d=\left(\frac{p}{3}\right)^3+\left(\frac{q}{2}\right)^2.$$

Таблица 3.1. Количество корней кубического уравнения

Дискриминант	**Количество действительных корней**	**Количество комплексных корней**
D≥0	1	2
D<0	3	-

Корни приведенного уравнения могут быть рассчитаны по формулам Кардано:

$$y_1=u+v$$

$$y_2=\frac{-u+v}{2}+\frac{u-v}{2}i\sqrt{3} \qquad (3.4)$$

$$y_3=\frac{-u+v}{2}-\frac{u-v}{2}i\sqrt{3}$$

где

$u=\sqrt[3]{\frac{-q}{2}+\sqrt{D}}$, $v=\sqrt[3]{\frac{-q}{2}-\sqrt{D}}$.

При отрицательном дискриминанте уравнение (3.1) имеет три действительных корня, но они будут вычисляться через вспомогательные комплексные величины. Чтобы избавиться от этого, можно воспользоваться формулами:

$y_1=2\sqrt[3]{\rho}\cos(\frac{\varphi}{3})$

$y_2=2\sqrt[3]{\rho}\cos(\frac{\varphi}{3}+\frac{2\pi}{3})$ (3.5.)

$y_3 = 2\sqrt[3]{\rho}\cos(\frac{\varphi}{3} + \frac{4\pi}{3})$

Где $\rho = \sqrt{\frac{-p^3}{27}}$, $\cos(\varphi) = \frac{-q}{2\rho}$.

Таким образом, при положительном дискриминанте кубического уравнения (3.3) расчет корней будем вести по формулам (3.4), а при отрицательном – по формулам (3.5). После расчета корней приведенного уравнения (3.3) по формулам (3.4) или (3.5) необходимо по формулам

$x_k = y_k - \frac{r}{3}, \quad k = 1,2,3 \ldots$

перейти к корням заданного кубического уравнения (3.1).

Блок-схема решения кубического уравнения представлена на рис. 3.19.

Описание блок-схемы. В блоке 1 вводятся коэффициенты кубического уравнения, в блоках 2-3 рассчитываются коэффициенты канонического и приведенного уравнений. Блок 4 предназначен для вычисления дискриминанта. В блоке 5 проверяется знак дискриминанта кубического уравнения. Если он отрицателен, то корни вычисляются по формулам (3.5) (блоки 6-7). При положительном значении дискриминанта расчет идет по формулам (3.4) (блок 9, 10). Блоки 8 и 11 предназначены для вывода результатов на экран.

Текст программы с комментариями приведен ниже.

При расчете величин u и v в программе предусмотрена проверка значения подкоренного выражения.

Если $\frac{-q}{2} \pm \sqrt{D} > 0$, то $u = \sqrt[3]{\frac{-q}{2} + \sqrt{D}}$, а $v = \sqrt[3]{\frac{-q}{2} - \sqrt{D}}$

Если $\frac{-q}{2} \pm \sqrt{D} < 0$, то $u = \sqrt[3]{|\frac{-q}{2} + \sqrt{D}|}$, а $v = \sqrt[3]{|\frac{-q}{2} - \sqrt{D}|}$

Соответственно, при нулевом значении подкоренного выражения u и v обращаются в ноль.

```
#include <iostream>
#include <math.h>
using namespace std;
#define pi 3.14159  //Определение константы
int main()
{ float a,b,c,d,D,r,s,t,p,q,ro,fi,x1,x2,x3,u,v,h,g;
//Ввод коэффициентов
//кубического уравнения.
cout<<"a="; cin>>a;
cout<<"b="; cin>>b;
cout<<"c="; cin>>c;
cout<<"d="; cin>>d;
//Расчет коэффициентов
//канонического уравнения по формуле (3.2)
r=b/a; s=c/a; t=d/a;
```

```
//Вычисление коэффициентов
//приведенного уравнения по формуле (3.3)
p=(3*s-r*r)/3; q=2*r*r*r/27-r*s/3+t;
//Вычисление дискриминанта
//кубического уравнения
D=(p/3)*(p/3)*(p/3)+(q/2)*(q/2);
if (D<0)
{
//Формулы (3.5)
ro=sqrt((float)(-p*p*p/27));
fi=-q/(2*ro);
fi=pi/2-atan(fi/sqrt(1-fi*fi));
x1=2*pow(ro,(float)1/3)*cos(fi/3)-r/3;
x2=2*pow(ro,(float)1/3)*cos(fi/3+2*pi/3)-r/3;
x3=2*pow(ro,(float)1/3)*cos(fi/3+4*pi/3)-r/3;
cout<<"\n x1="<<x1<<"\t x2="<<x2;
cout<<"\t x3="<<x3<<"\n";
}
 else
{
//Формулы (3.4)
if (-q/2+sqrt(D)>0) u=pow((-q/2+sqrt(D)),(float)1/3);
else
if (-q/2+sqrt(D)<0) u=-pow(fabs(-
q/2+sqrt(D)),(float)1/3);
else u=0;
if (-q/2-sqrt(D)>0) v=pow((-q/2-sqrt(D)),(float)1/3);
else
if (-q/2-sqrt(D)<0) v=-pow(fabs(-q/2-
sqrt(D)),(float)1/3);
else v=0;
x1=u+v-r/3; //Вычисление действительного
//корня кубического уравнения.
h=-(u+v)/2-r/3; //Вычисление действительной
g=(u-v)/2*sqrt((float)3); //и мнимой части комплексных
//корней
cout<<"\n x1="<<x1;
cout<<"\t x2="<<h<<"+i*("<<g<<")";
cout<<"\t x3="<<h<<"-i*("<<g<<") \n";
}
return 0;
}
```

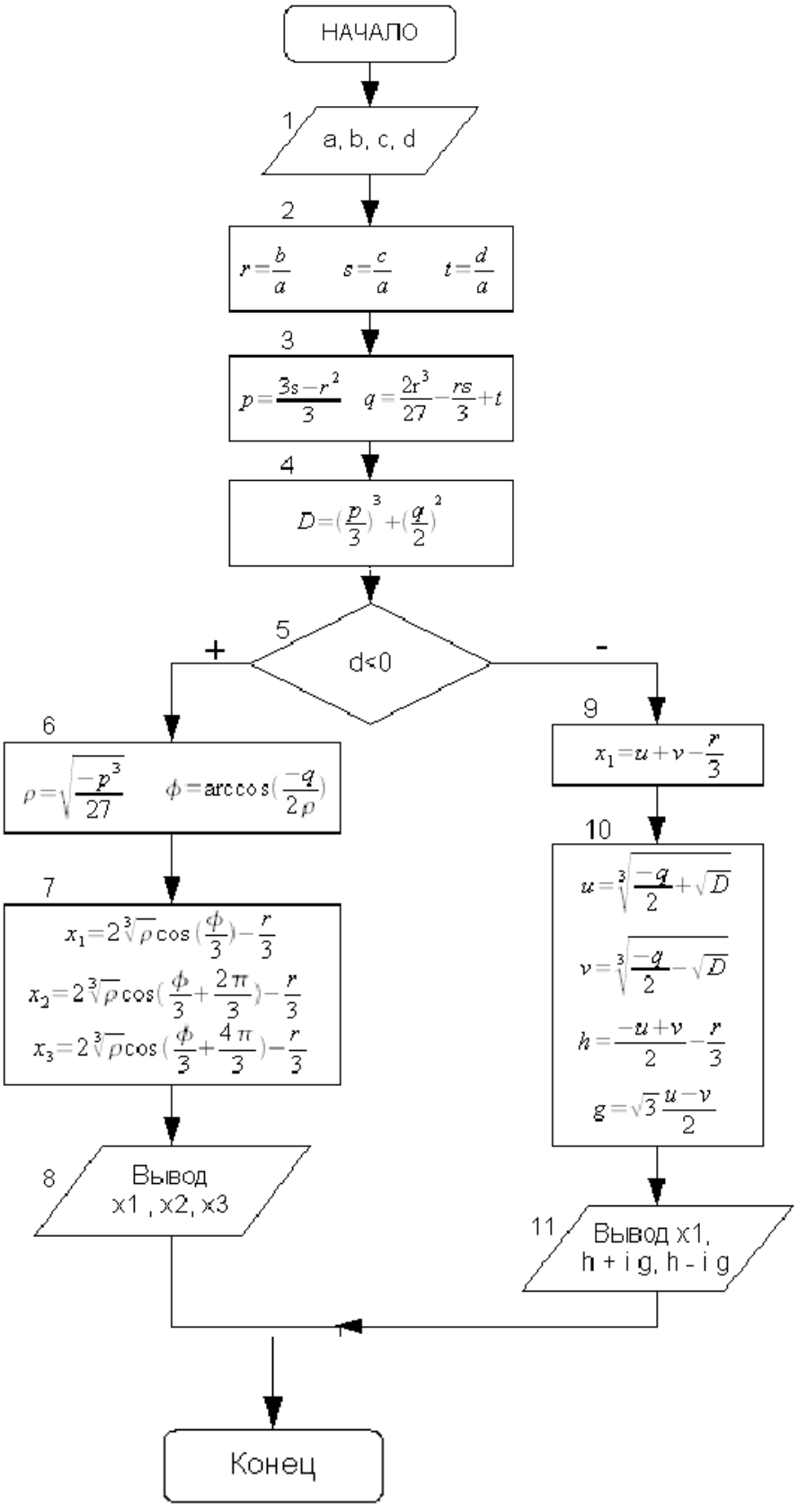


Рис. 11. Алгоритм решения кубического уравнения

Пример 9. *Заданы коэффициенты a, b и c биквадратного уравнения* $ax^4+bx^2+c=0$. *Найти все его действительные корни.*

Входные данные: a, b, c.

Выходные данные: x1, x2, x3, x4.

Для решения биквадратного уравнения необходимо заменой $y=x^2$ привести его к квадратному уравнению $ay^2+by+c=0$ и решить это уравнение.

Опишем алгоритм решения этой задачи (рис. 3.20):

1. Ввод коэффициентов биквадратного уравнения a, b и c (блок 1).
2. Вычисление дискриминанта уравнения d (блок 2).
3. Если d < 0 (блок 3), вывод сообщения, что корней нет (блок 4), а иначе определяются корни соответствующего квадратного уравнения y1 и y2 (блок 5).
4. Если y1 и y2 < 0 (блок 6) , то вывод сообщения, что корней нет (блок 7).
5. Если y1 и y2 ≥ 0 (блок 8), то вычисляются четыре корня по формулам $\pm\sqrt{y_1}, \pm\sqrt{y_2}$, (блок 9) и выводятся значения корней (блок 10).
6. Если условия 4) и 5) не выполняются, то необходимо проверить знак y1. Если y1≥ 0 (блок 11), то вычисляются два корня по формуле $\pm\sqrt{y_1}$, (блок 12), иначе (если y2≥ 0) вычисляются два корня по формуле $\pm\sqrt{y_2}$ (блок 13).

Вывод вычисленных значений корней (блок 14).

Внимание! Если в условном операторе проверяется двойное условие необходимо применять логические операции ||, &&, !. Например, условие «если y1 и y2 положительны» правильно записать так: if (y1>=0 && y2>=0)

Текст программы решения биквадратного уравнения приведен ниже.

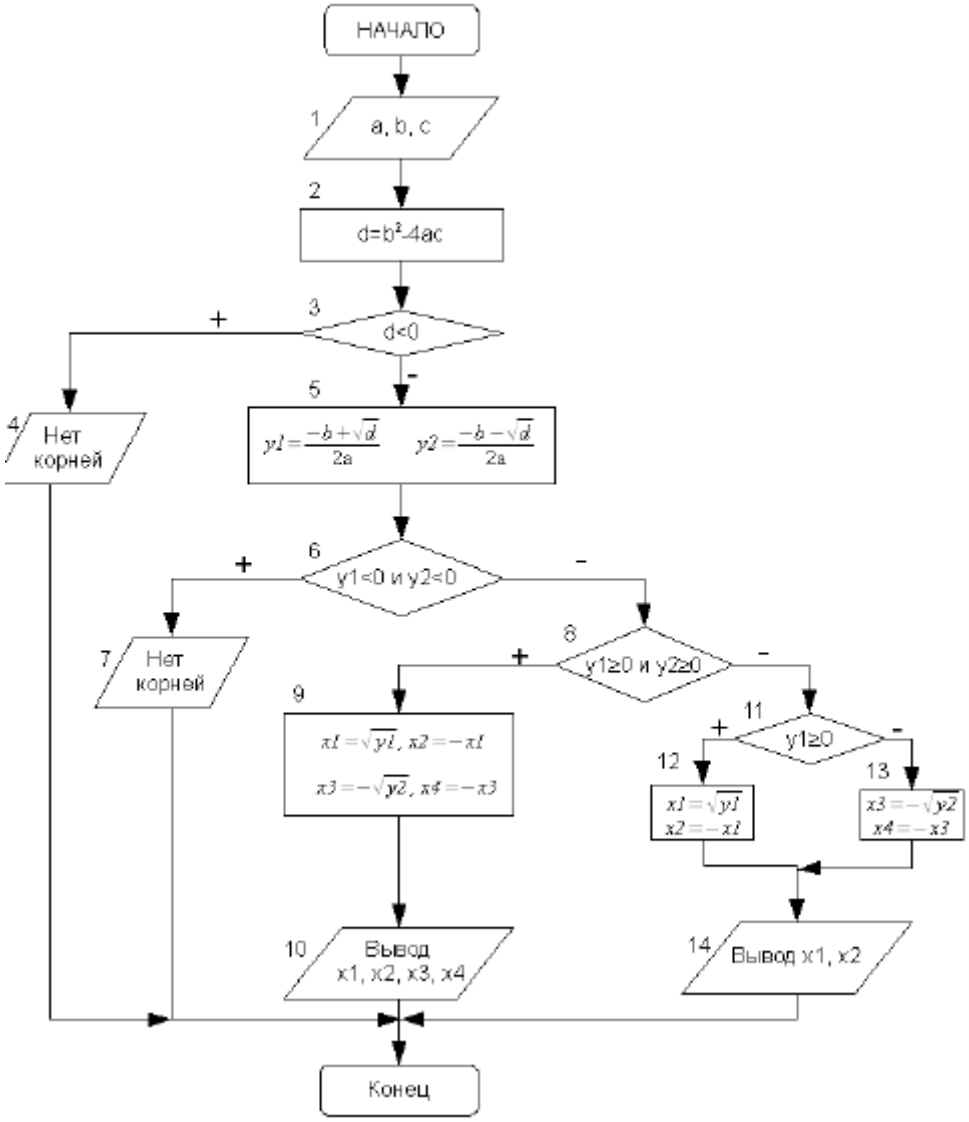


Рис. 12. Алгоритм решения биквадратного уравнения

```
#include <iostream.h
>
#include <math.h>
using namespace std;
int main()
{
//Описание переменных:
//a,b,c – коэффициенты биквадратного уравнения,
//d – дискриминант,
//x1,x2,x3,x4 – корни биквадратного уравнения,
//y1,y2 – корни квадратного уравнения ay^2+by+c=0,
float a,b,c,d,x1,x2,x3,x4,y1,y2;
//Ввод коэффициентов уравнения.
cout<<"a="; cin>>a;
cout<<"b="; cin>>b;
```

```
cout<<"c="; cin>>c;
d=b*b-4*a*c; //Вычисление дискриминанта.
if (d<0) //Если дискриминант отрицательный,
//вывод сообщения «Корней нет».
cout<<"Real roots are not present \n";
else //Если дискриминант положительный,
{
//Вычисление корней
//соответствующего
//квадратного уравнения.
y1=(-b+sqrt(d))/2/a;
y2=(-b-sqrt(d))/(2*a);
//Если оба корня
//квадратного уравнения
//отрицательные,
if (y1<0 && y2<0)
//вывод сообщения
//«Корней нет»
cout<<"Real roots are not present \n";
//Если оба корня
//квадратного уравнения
//положительные,
else if (y1>=0 && y2>=0)
{
//Вычисление четырех
//корней биквадратного
//уравнения
x1=sqrt(y1);
x2=-x1;
x3=sqrt(y2);
x4=-sqrt(y2);
//Вывод корней
//уравнения на экран.
cout<<"\t X1="<<x1<<"\t X2="<<x2;
cout<<"\t X3="<<x3<<"\t X4="<<x4<<"\n";
}
//Если не выполнились условия
//1. y1<0 И y2<0
//2. y1>=0 И y2>=0,
//то проверяем условие y1>=0.
else if (y1>=0)
//Если оно истинно
{
//вычисляем два корня
//биквадратного уравнения.
```

```
x1=sqrt(y1);
x2=-x1;
cout<<"X1="<<x1<<"\t X2="<<x2<<"\n";
}
else
//Если условие y1>=0 ложно, то
{
//вычисляем два корня
//биквадратного уравнения
x1=sqrt(y2);
x2=-x1;
cout<<"X1="<<x1<<"\t X2="<<x2<<"\n";
}
}
return 0;
}
```

2.2. Оператор switch

Оператор `switch` (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Структурная схема оператора приведена на рис. 4.

Формат оператора:

```
                switch ( выражение ) {
case константное_ выражение_1: [ список_ операторов_1]
case константное_ выражение_2: [ список_ операторов_2]
…
case константное_ выражение_n: [ список_ операторов_n]
               [default : операторы]}
```

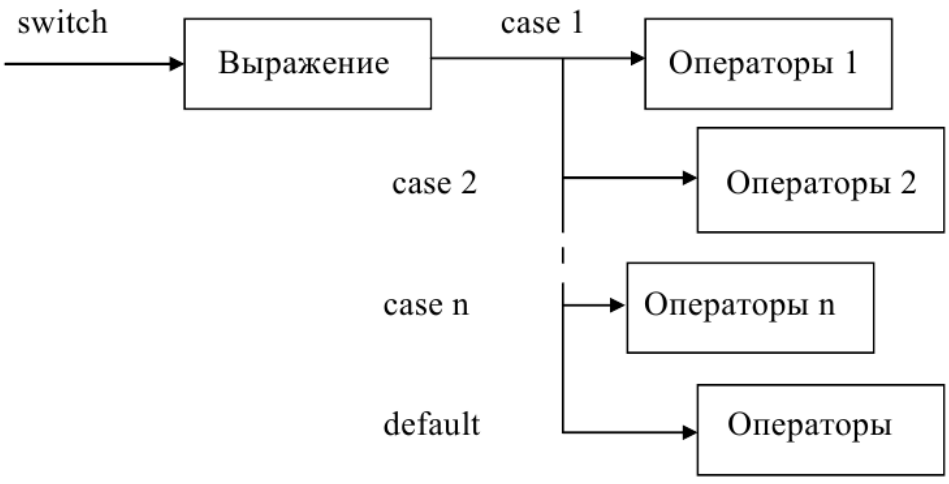


Рис. 13. Структурная схема оператора switch

Выполнение оператора начинается с вычисления выражения (оно должно быть целочисленным), а затем управление передается первому оператору из списка, помеченного константным выражением, значение которого совпало с вычисленным. После этого, если выход из переключателя явно не указан, последовательно выполняются все остальные ветви.

Выход из переключателя обычно выполняется с помощью операторов `break` или `return`. Оператор `break` выполняет выход из самого внутреннего из объемлющего его оператора `switch`. Оператор `return` выполняет выход из функции, в теле которой он записан.

Все константные выражения должны иметь разные значения, но быть одного и того же целочисленного типа. Несколько меток могут следовать подряд. Если совпадения не произошло, выполняются операторы, расположенные после слова `default` (а при его отсутствии управление передается следующему за `switch` оператору).

Итак, решая задачи этого раздела, следует помнить, что:

- оператор switch предназначен для выбора одного из нескольких возможных направлений дальнейшего хода программы;
- выбор последовательности выполнения осуществляется в зависимости от равенства значения переменной-селектора константе, указанной после слова case;
- если значение переменной селектора не равно ни одной из констант, написанных после слова case, то выполняется инструкции, расположенные после слова default ;
- в качестве переменной-селектора можно использовать переменную целого (int) или символьного (char) типа.

2.3. Операторы передачи управления

Операторы передачи управления принудительно изменяют порядок выполнения команд. В C++ таких операторов четыре: `goto`, `break`, `continue` и `return`.

Оператор `goto метка`, где метка обычный идентификатор, применяют для безусловного перехода, он передает управление оператору с меткой:

```
метка: оператор;
```

Оператор `break` осуществляет немедленный выход из циклов `while`, `do…while` и `for`, а так же из оператора выбора `switch`. Управление передается оператору, находящемуся непосредственно за циклом или оператором выбора.

Оператор `continue` начинает новую итерацию цикла, даже если предыдущая не была завершена.

Оператор `return` выражение завершает выполнение функции и передает управление в точку ее вызова. Если функция возвращает значение типа `void`, то выражение в записи оператора отсутствует. В противном случае выражение должно иметь скалярный тип.

Пример 10. *Программа переводит числовую оценку ученика в ее словесный эквивалент (5 – «отлично», 4 – «хорошо», 3 – «удовлетворительно», 2 – «неудовлетворительно»).*

```
#include <iostream.h>
    void main()
    {  int ball;
       cout<<"\nOcenka:"; cin>> ball;
       switch (ball)
     {     case 2: cout<<"\tNeud!\n"; break;
        case 3: cout<<"\tUd!\n"; break;
        case 4: cout<<"\tGood!\n"; break;
        case 5: cout<<"\tExellent!\n"; break;
        default: cout<<"\tNo!\n";}}
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

Задание 1. Составить алгоритм и написать программу на языке С++ решения задачи согласно своему варианту.

Варианты заданий

1. Задана точка М с координатами (x,y). Определить месторасположение этой точки в декартовой системе координат (является ли эта точка началом координат, лежит на одной из координатных осей или расположена в одном из координатных углов).
2. Задана квадратичная функция вида y=ax2+bx+c. Вывести сообщения, как направлены ветви параболы, сколько у нее точек пересечения с осью OX.
3. Задан параллелограмм со сторонами a, b и углом α между ними. Определить тип параллелограмма (ромб, прямоугольник или квадрат), если это возможно.
4. Известны углы α и β у основания трапеции. Выяснить, если это возможно, тип трапеции (прямоугольная, равнобедренная, прямоугольник).
5. Задан круг с центром в точке $O(x_0, y_0)$ и радиусом R_0 и точка $A(x_1, y_1)$. Определить месторасположение точки по отношению к кругу (находится внутри круга, вне его или лежит на окружности).
6. Определите, пересекаются ли парабола y=cx2+dx+f и прямая y=ax+b. При положительном ответе найти точки пересечения.
7. Заданы три функции $y1=x^3$, $y2=x^3+1$, $y3 = \frac{1}{1+x^2}$. Определить, являются ли эти функции четными или нечетными.
8. Выяснить, пересекаются ли параболы $y=ax^2+bx+c$ и $y=dx^2+ex + f$. При положительном ответе найти точки пересечения.

9. Выяснить, пересекаются ли кривые $y=ax^3+bx^2+cx+d$ и $y=ex^3+fx^2+gx+h$. При положительном ответе найти точки пересечения.
10. Определите, пересекаются ли кривая $y=ax^3+bx^2+cx+d$ и прямая $y=fx+g$. При положительном ответе найти точки пересечения.
11. Задана окружность с центром в точке $O(x_0,y_0)$ и радиусом R_0 и прямая $y=ax+b$. Определить, пересекаются ли прямая и окружность. При положительном ответе найти точки пересечения.
12. Заданы две окружности: с центром в точке $O(x_0, y_0)$ и радиусом R_0 и с центром в точке $O(x_1, y_1)$ и радиусом R_1. Определите, во скольких точках пересекаются окружности.
13. Заданы три точки на плоскости: M с координатами (x_1,y_1), L с координатами (x_2,y_2) и H с координатами (x_3,y_3). Определите, лежат ли они на одной прямой. При отрицательном ответе найти площадь и периметр треугольника MLH.
14. Заданы три точки $A(a_1,a_2,a_3)$, $B(b_1,b_2,b_3)$ и $C(c_1,c_2,c_3)$. Определить, между какими точками расстояние будет наименьшим.
15. Задан треугольник с углами α, β и γ. Определить тип треугольника — остроугольный, прямоугольный или тупоугольный.
16. Заданы точки $A(a_1,a_2)$ и $B(b_1,b_2)$. Определить, лежат ли они на прямой $y=ax+b$.
17. Известны уравнения двух прямых $y=a_1x+b_1$ и $y=a_2x+b_2$. Определить, являются ли эти прямые параллельными или перпендикулярными, если нет, то найти угол между ними.
18. Задан треугольник со сторонами a, b и c. Определить, является ли этот треугольник равносторонним, равнобедренным, если нет, вычислить площадь треугольника.
19. Даны уравнения двух прямых $y=a_1x+b_1$ и $y=a_2x+b_2$. Определить, пересекаются ли эти прямые, совпадают или параллельны.
20. Даны 3 дроби $\frac{a_1}{b_1}, \frac{a_2}{b_2}, \frac{a_3}{b_3}$. . Найти, какая из трех дробей наибольшая.
21. Определить, имеет ли решение система $\begin{cases} ax + by = c \\ dx + ey = f \end{cases}$. Если решение есть, найти значение x и y.
22. Определить, при каких значениях x и y векторы $A=a_1i+a_2j+xk$ и $B=yi+b_2j+b_3k$ коллинеарны и какой из векторов короче.
23. Проверить коллинеарность векторов $A=(a_1, a_2, a_3)$ и $B=(b_1, b_2, b_3)$. Установить, какой из них длиннее и во сколько раз.
24. Даны координаты вершин двух треугольников ABC и DFG: $A(a_1,a_2)$, $B(b_1,b_2)$, $C(c_1,c_2)$, $D(d_1,d_2)$, $F(f_1,f_2)$, $G(g_1,g_2)$. Определить, периметр какого треугольника больше.
25. Даны две прямые $y=a_1\cdot x+c_1$ и $y=a_2\cdot x+c_2$. Определить условие перпендикулярности прямых, и если оно не выполнятся, найти угол между н и м и.

26. Задана показательная функция $y=a^x$. Проверить, является ли функция возрастающей (при $a>1$)или убывающей (при $0 \le a \le 1$). Задана функция

обратной пропорциональности $y = \frac{k}{x}$. Определить, в каких координатных углах расположены ветви гиперболы.

Задание 2. Составить алгоритм и написать программу на языке C++ решения задачи с использованием оператора switch согласно своему варианту.

Варианты заданий

1. Составить программу, которая в зависимости от порядкового номера дня недели (1,2, …, 7) выводит на экран его название (понедельник, вторник, …, воскресенье).
2. Составить программу, которая в зависимости от порядкового номера дня месяца (1,2, …, 12) выводит на экран его название (январь, февраль, …, декабрь).
3. Составить программу, которая в зависимости от порядкового номера дня месяца (1,2, …, 12) выводит на экран время года, к которому относится этот месяц (январь, февраль, …, декабрь).
4. Составить программу, которая в зависимости от порядкового номера месяца (1,2, …, 12) выводит на экран количество дней в этом месяце (рассмотреть случай, когда год не является високосным).
5. Дано целое число k ($1 \le k \le 365$). Определить, каким днем недели (понедельником, вторником, …, субботой или воскресеньем) является k-й день невисокосного года, в котором 1 января – понедельник.
6. С начала 1990 года по некоторый день прошло n месяцев и 2 дня. Определить название месяца (январь, февраль, …) этого дня.
7. Дата некоторого дня характеризуется двумя натуральными числами: m – порядковый номер месяца, n – число. По заданным m и n определить дату предыдущего дня (принять, что m и n не характеризуют 1 января, год не является високосным).
8. Дата некоторого дня характеризуется двумя натуральными числами: m – порядковый номер месяца, n – число. По заданным m и n определить дату следующего дня (принять, что m и n не характеризуют 31 декабря, год не является високосным).

Задание 3. ***Разветвляющийся процесс. Вычисление значения функции*.**

3.1. Разработать программу на языке C++. Дано вещественное число a. Для функции y=f(x), график которой приведен ниже вычислить f(a). Варианты заданий представлены на рис. 4.1 - 4.25.

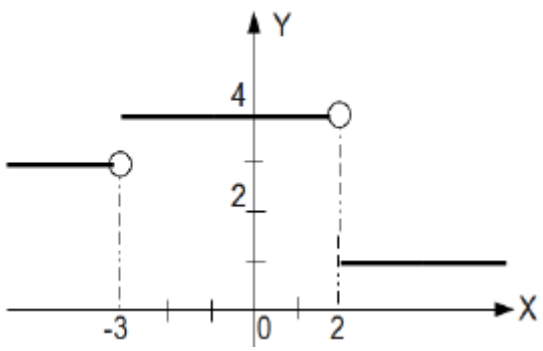


Рис. 4.1. Вариант 1

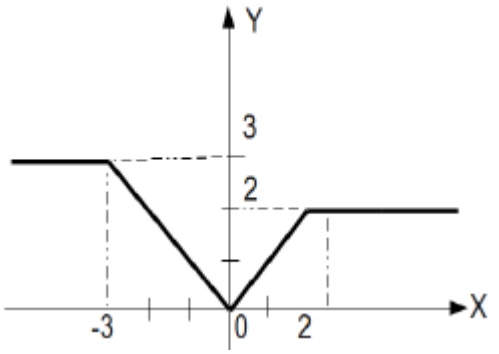


Рис. 4.2. Вариант 2

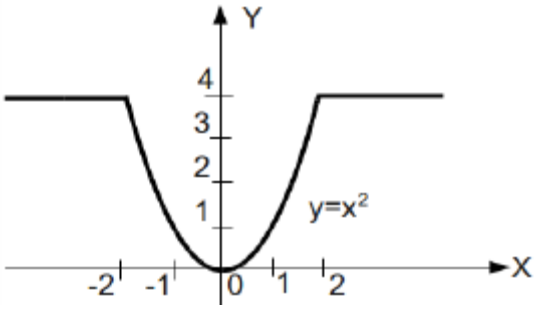


Рис. 4.3. Вариант 3

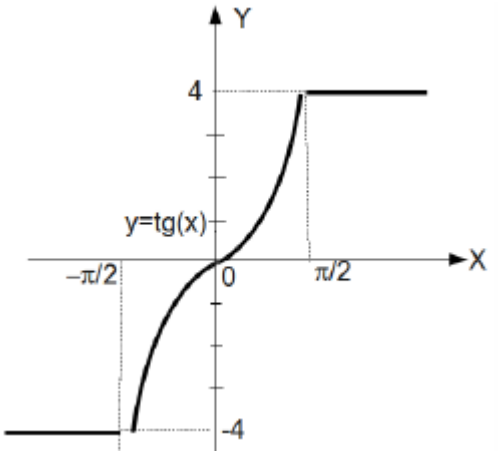


Рис. 4.4. Вариант 4

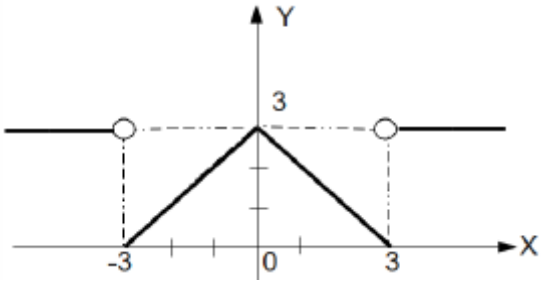


Рис. 4.5. Вариант 5

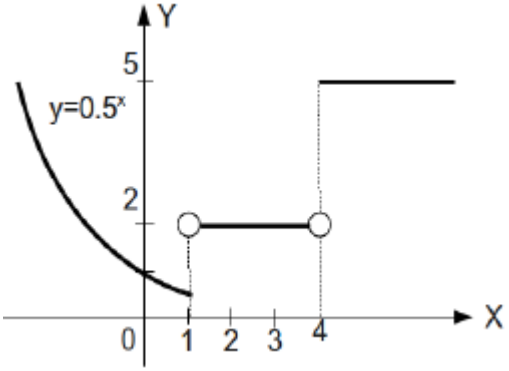


Рис. 4.6. Вариант 6

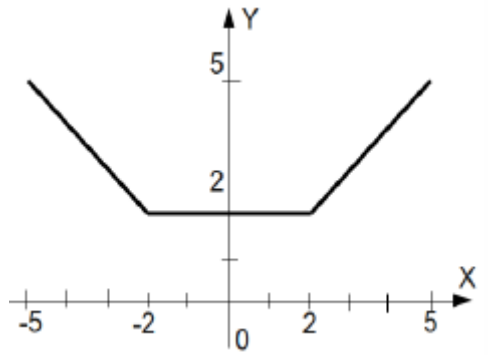


Рис. 4.7. Вариант 7

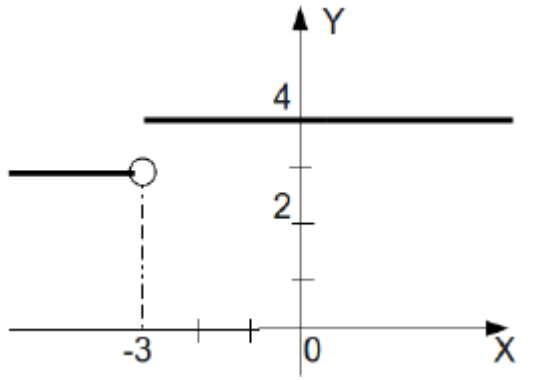


Рис. 4.8. Вариант 8

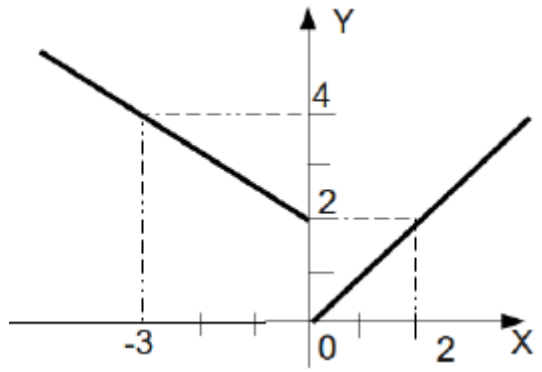


Рис. 4.9. Вариант 9

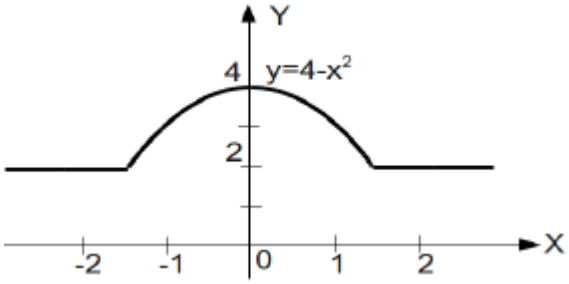


Рис. 4.10. Вариант 10

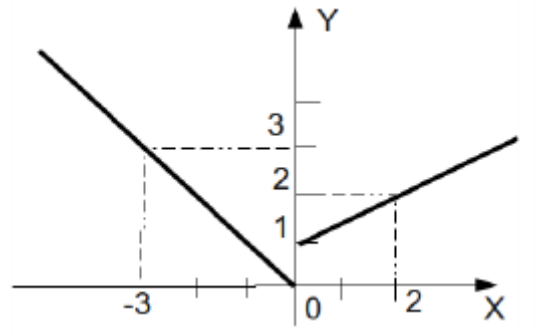


Рис. 4.11. Вариант 11

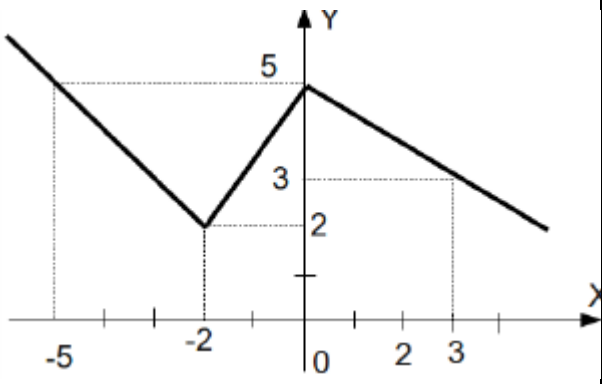


Рис. 4.12. Вариант 12

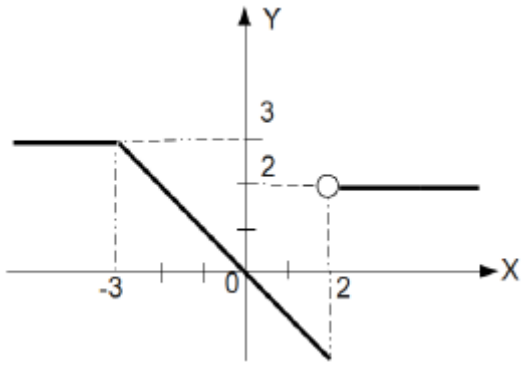


Рис. 4.13. Вариант 13

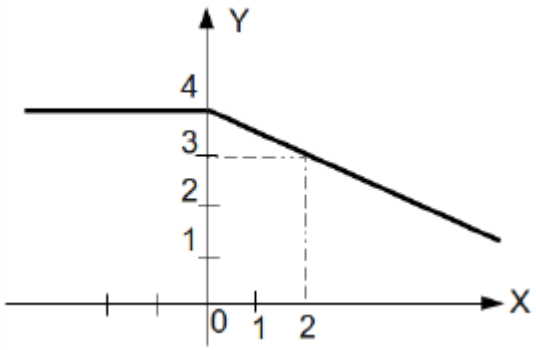


Рис. 4.14 Вариант 14

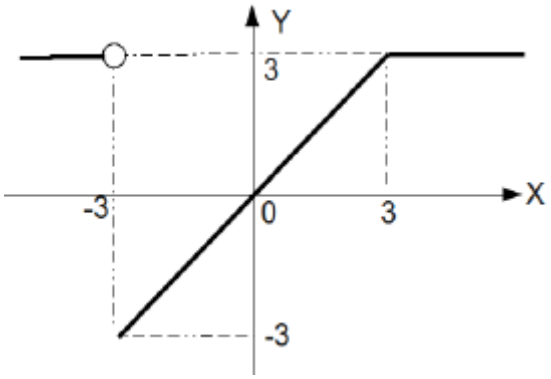


Рис. 4.15. Вариант 15

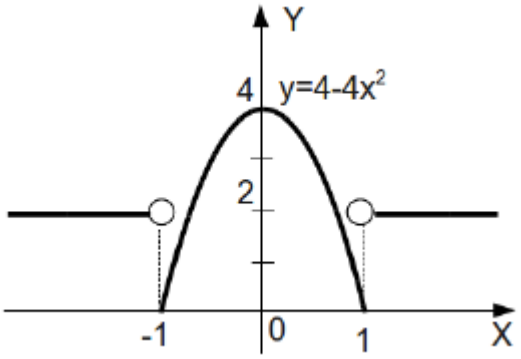


Рис. 4.16. Вариант 16

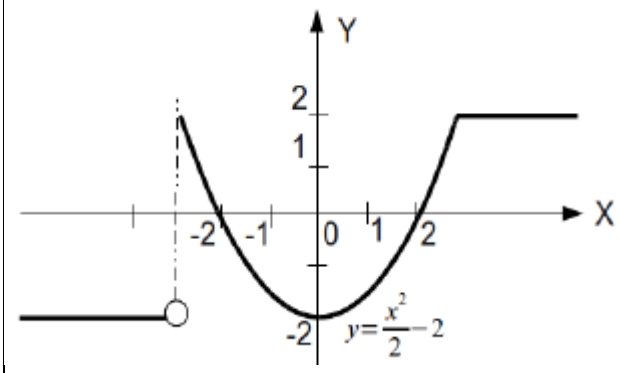


Рис. 4.17. Вариант 17

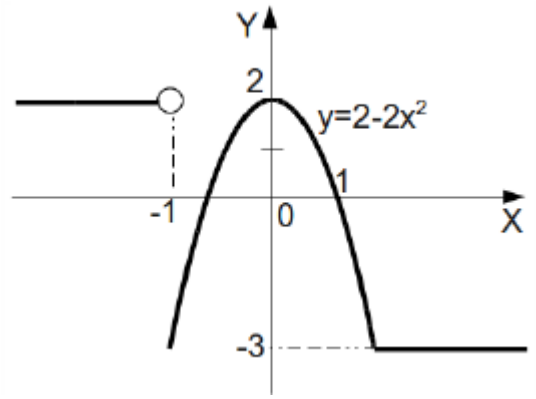


Рис. 4.18. Вариант 18

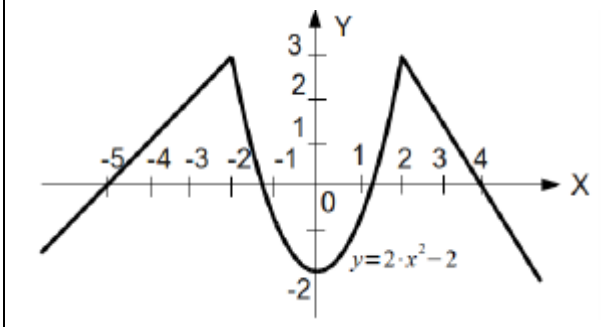


Рис. 4.19. Вариант 19

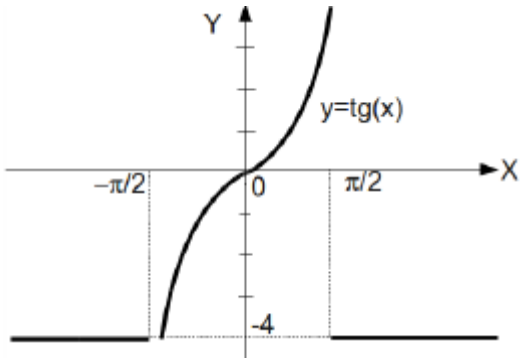


Рис. 4.20. Вариант 20

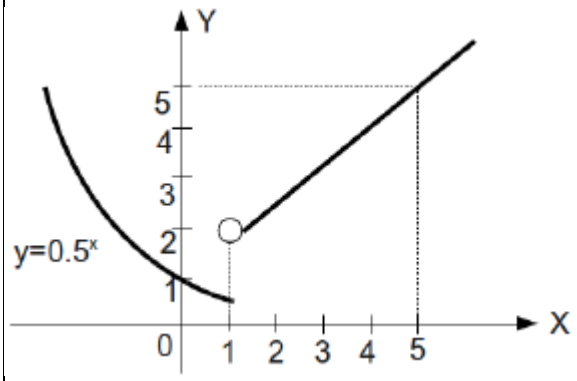


Рис. 4.21. Вариант 21

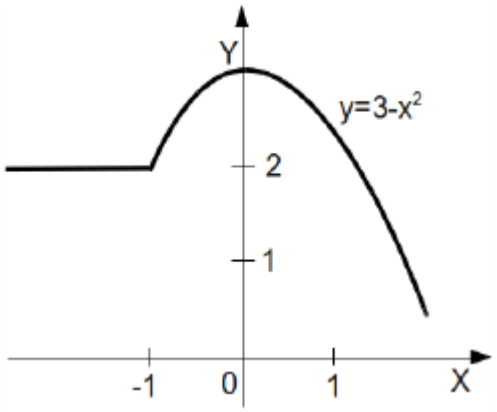
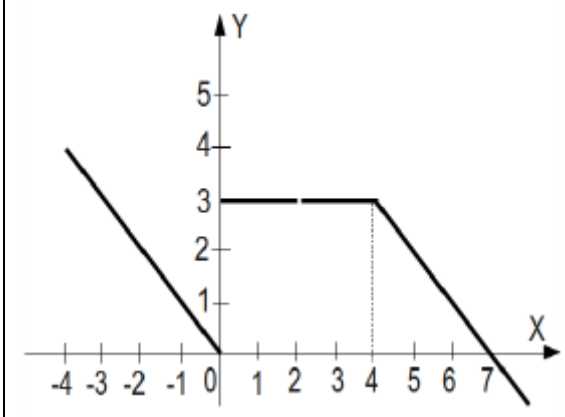
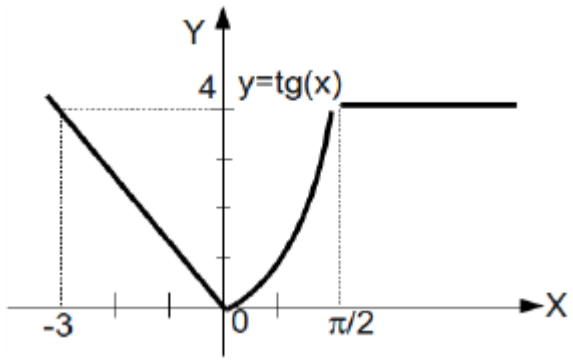


Рис. 4.22. Вариант 22





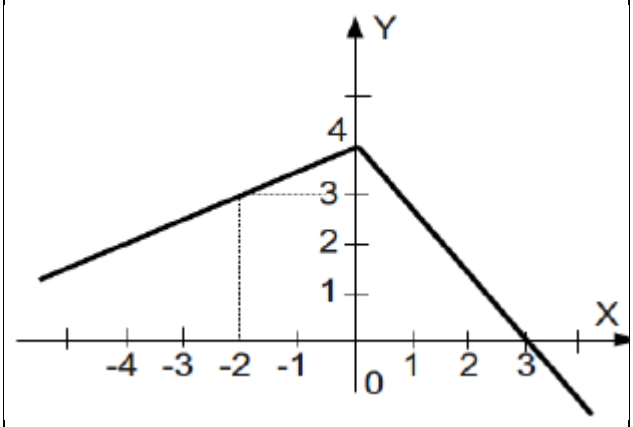


Рис. 4.23. Вариант 23

Рис. 4.24. Вариант 24

Рис. 4.25. Вариант 25

Разветвляющийся процесс. Попадание точки в плоскость

3.2. Разработать программу на языке C++. Даны вещественные числа x и y. Определить принадлежит ли точка с координатами (x; y) заштрихованной части плоскости. Варианты заданий представлены на рис. 3.64 - 3.88.

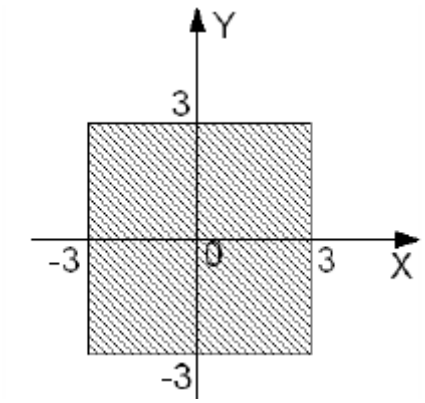


Рис. 5.1. Вариант 1

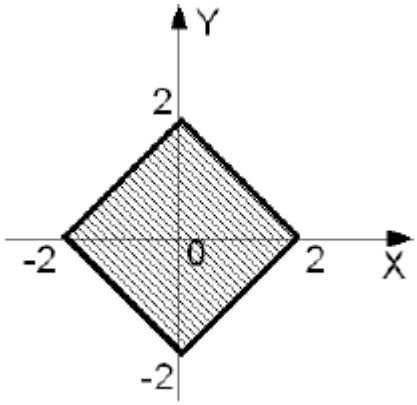


Рис. 5.2. Вариант 2

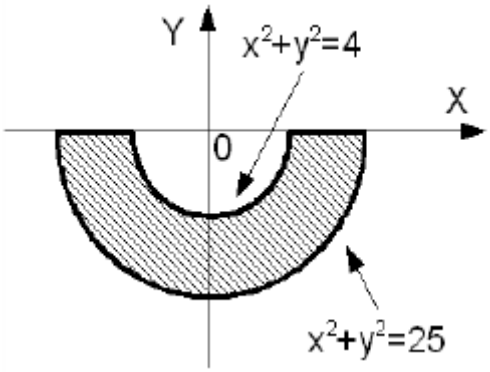


Рис. 5.3. Вариант 3

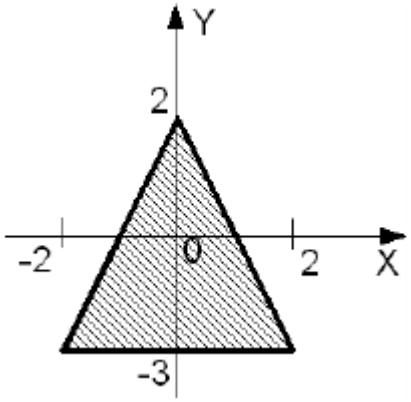


Рис. 5.4. Вариант 4

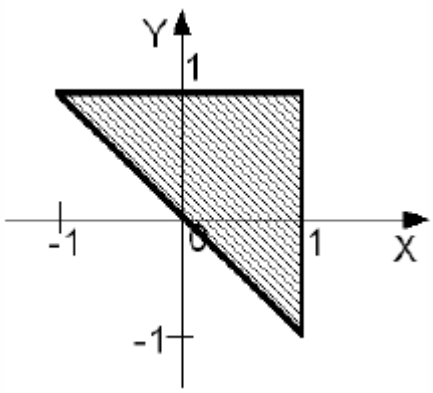


Рис. 5.5. Вариант 5

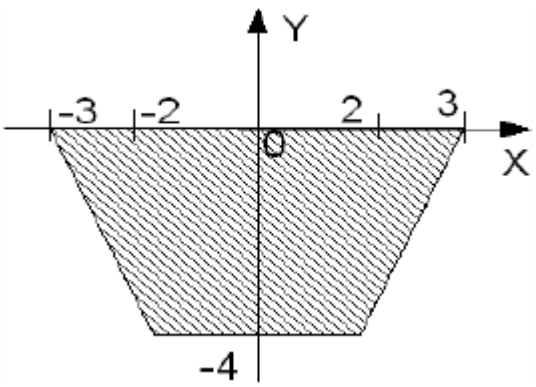


Рис. 5.6. Вариант 6

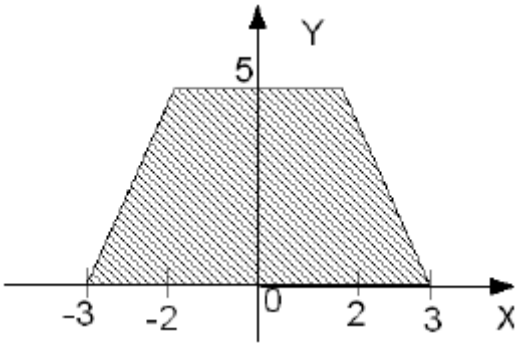


Рис. 5.7. Вариант 7

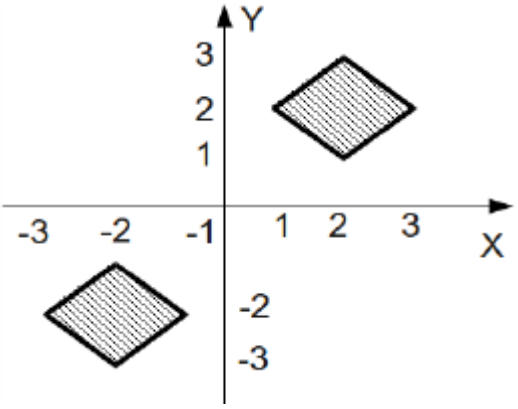


Рис. 5.8. Вариант 8

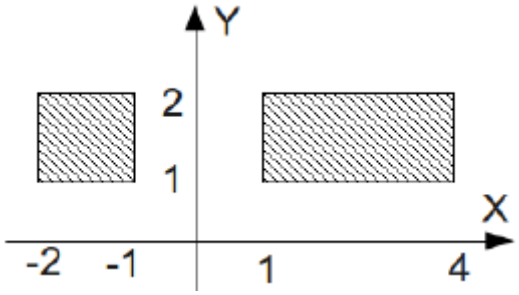


Рис. 5.9. Вариант 9

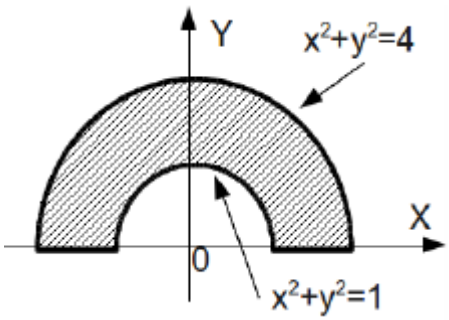


Рис. 5.10. Вариант 10

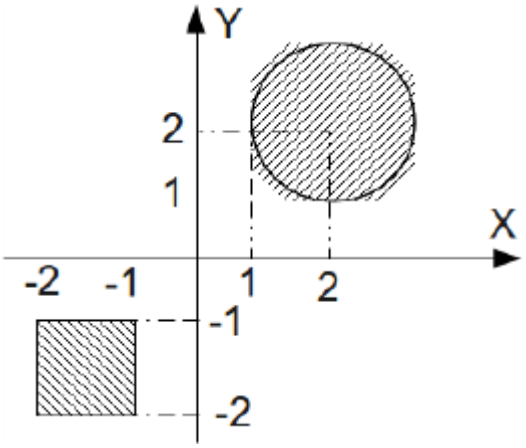


Рис. 5.11. Вариант 11

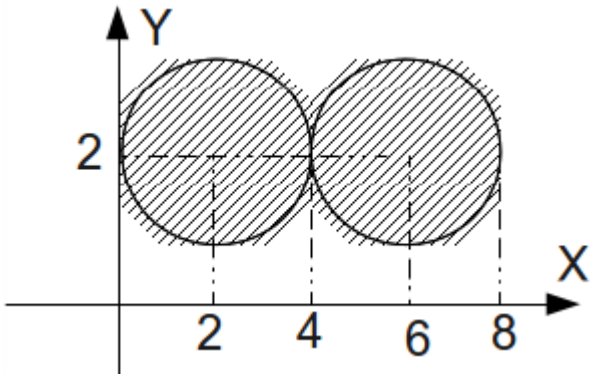


Рис. 5.12. Вариант 12

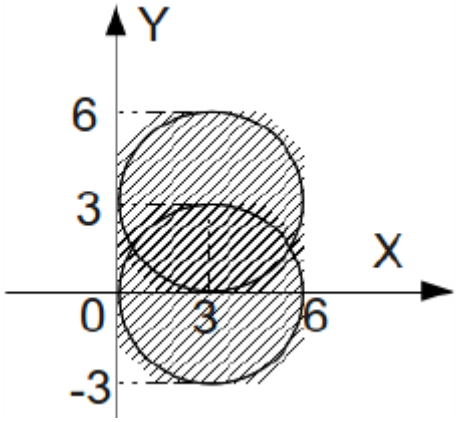


Рис. 5.13. Вариант 13

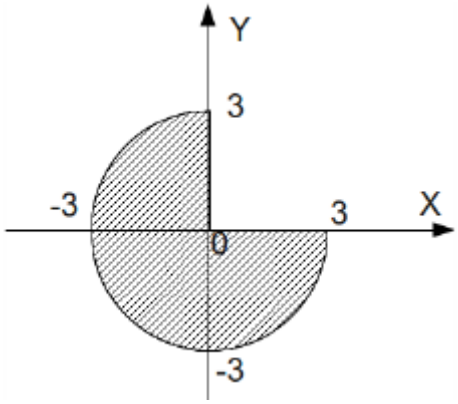


Рис. 5.14. Вариант 14

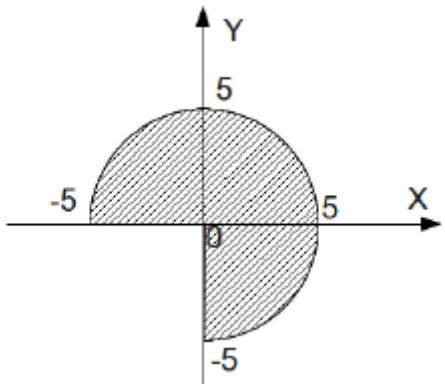


Рис. 5.15. Вариант 15

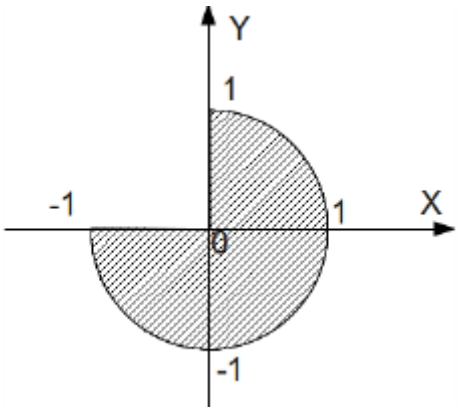


Рис. 5.16. Вариант 16

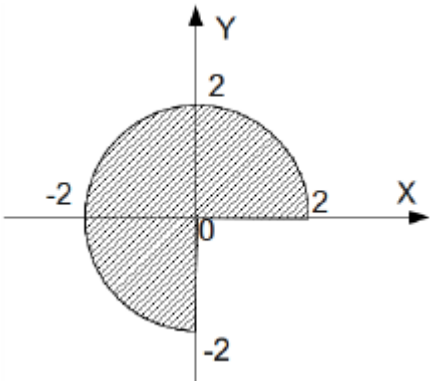


Рис. 5.17. Вариант 17

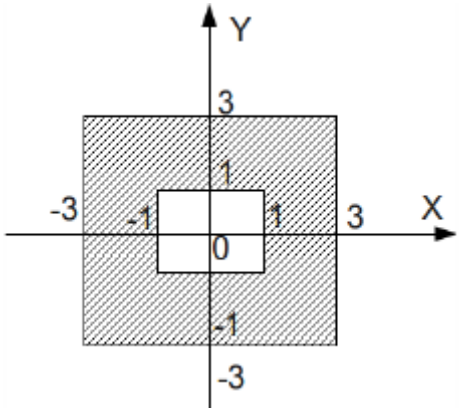


Рис. 5.18. Вариант 18

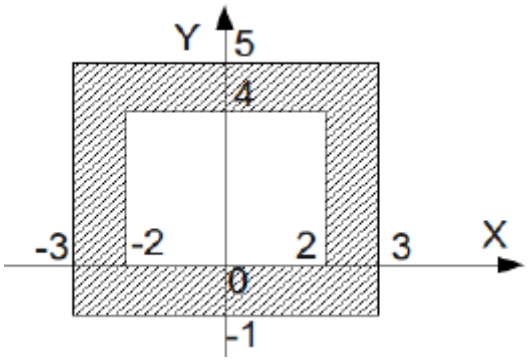


Рис. 5.19. Вариант 19

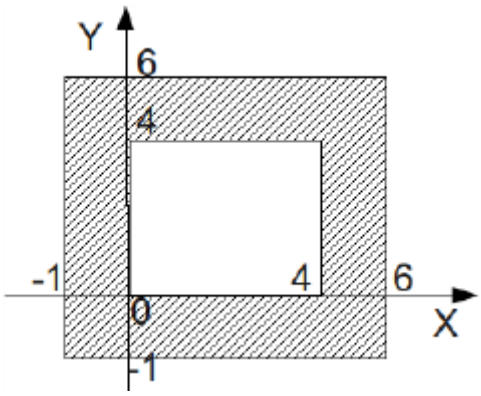


Рис. 5.20. Вариант 20

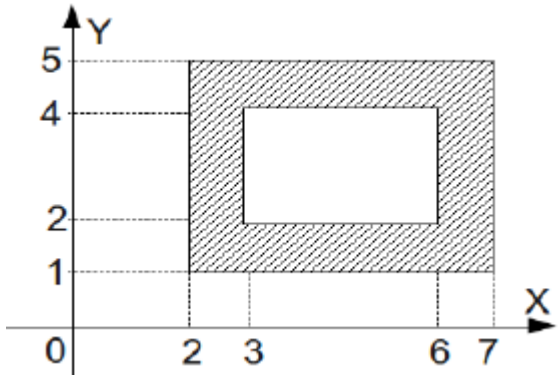


Рис. 5.21. Вариант 21

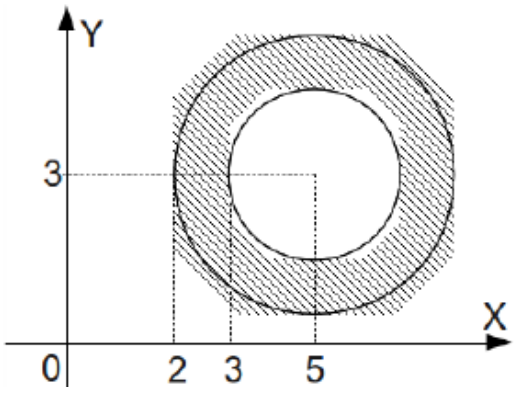


Рис. 5.22. Вариант 22

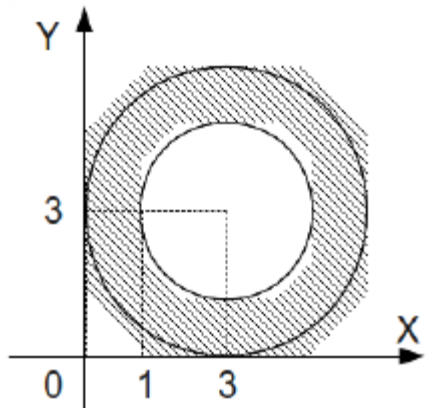


Рис. 5.23. Вариант 23

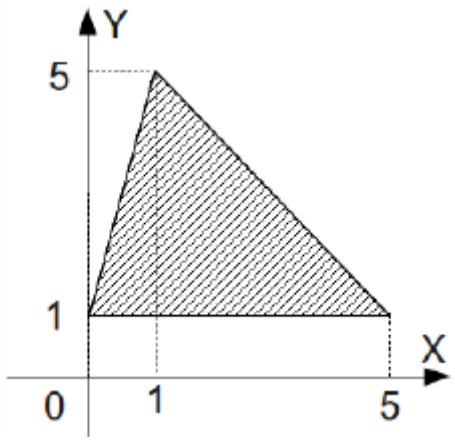


Рис. 5.24. Вариант 24

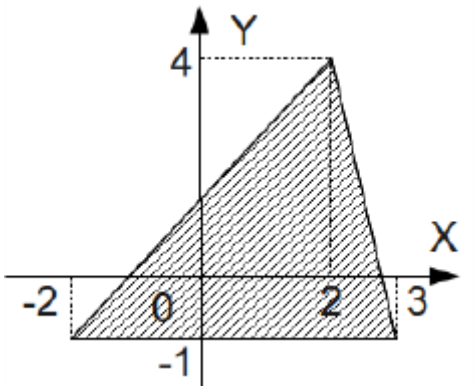


Рис. 5.25. Вариант 25

Разветвляющийся процесс. Пересечение линий и решение уравнений.

3.3. Разработать программу на языке C++ для следующих заданий:

1. Задан круг с центром в точке $O(x_0, y_0)$, радиусом R_0 и точка $A(x_1, y_1)$. Определить, находится ли точка внутри круга.
2. Задана окружность с центром в точке $O(x_0, y_0)$, радиусом R_0. Определить пересекается ли заданная окружность с осью абсцисс, если пересекается найти точки пересечения.
3. Задана окружность с центром в точке $O(x_0, y_0)$, радиусом R_0. Определить пересекается ли заданная окружность с осью ординат, если пересекается найти точки пересечения.
4. Задана окружность с центром в точке $O(x_0, y_0)$, радиусом R_0и прямая $y=ax+b$. Определить, пересекаются ли прямая и окружность. Если пересекаются, найти точки пересечения.
5. Заданы окружности. Первая с центром в точке $O(x_1, y_1)$ и радиусом R_1, вторая с центром в точке $O(x_2, y_2)$ и радиусом R_2. Определить пересекаются окружности, касаются или не пересекаются.
6. Заданы три точки $A(x_1,y_1)$, $B(x_2,y_2)$, $C(x_3,y_3)$. Определить какая из точек наиболее удалена от начала координат.

7. Заданы три точки $A(x_1,y_1)$, $B(x_2,y_2)$, $C(x_3,y_3)$. Определить какая из точек B или C наименее удалена от точки A.
8. Определить, пересекаются ли линии $y=ax+b$ и $y=kx+m$. Если пересекаются, найти точку пересечения.
9. Определить, пересекает ли линия $y=ax+b$ ось абсцисс. Если пересекает, найти точку пересечения.
10. Определить, пересекаются ли линии $y=ax^3+bx^2+cx+d$ и $y=kx+m$. Если пересекаются, найти точки пересечения.
11. Определить, пересекаются ли линии $y=ax^3+bx^2+cx+d$ и $y=kx^3+mx^2+nx+p$. Если пересекаются, найти точки пересечения.
12. Определить, пересекаются ли линии $y=ax^3+bx^2+cx+d$ и $y=ax^3+mx^2+nx+p$. Если пересекаются, найти точки пересечения.
13. Определить, пересекаются ли линии $y=ax^3+bx^2+cx+d$ и $y=mx^2+nx+p$. Если пересекаются, найти точку пересечения.
14. Определить, пересекает ли линия $y=ax^3+bx^2+cx+d$ ось абсцисс. Если пересекает, найти точку пересечения.
15. Определить, пересекаются ли параболы $y=ax^2+bx+c$ и $y=dx^2+mx+n$. Если пересекаются, то найти точки пересечения.
16. Определить, пересекаются ли линии $y=bx^2+cx+d$ и $y=kx+m$. Если пересекаются, найти точки пересечения
17. Найти точки пересечения линии $y=ax^2+bx+c$ с осью абсцисс. Если линии не пересекаются выдать соответствующее сообщение.
18. Определить, пересекаются ли линии $y=ax^4+bx^3+cx^2+dx+f$ и $y=bx^3+mx^2+dx+p$. Если пересекаются, найти точки пересечения.
19. Определить, пересекаются ли линии $y=ax^4+bx^2+kx+c$ и $y=mx^2+kx+p$. Если пересекаются, найти точки пересечения.
20. Определить, пересекает ли линия $y=ax^4+bx^2+c$ ось абсцисс. Если пересекает, найти точки пересечения.
21. Найти комплексные корни уравнения $y=ax^4+bx^2+c$. Если в уравнении нет комплексных корней вывести соответствующее сообщение.
22. Найти комплексные корни уравнения $y=ax^3+bx^2+cx+d$. Если в уравнении нет комплексных корней вывести соответствующее сообщение.
23. Найти комплексные корни уравнения $y=ax^2+bx+c$. Если в уравнении нет комплексных корней вывести соответствующее сообщение.
24. Даны координаты точки на плоскости. Если точка совпадает с началом координат, то вывести 0. Если точка не совпадает с началом координат, но лежит на осиOX или OY, то вывести соответственно 1 или 2. Если точка не лежит на координатных осях, то вывести 3.
25. Даны координаты точки, не лежащей на координатных осях OX и OY. Определить номер координатной четверти, в которой находится данная точка.

ТЕМА 4. ПРОГРАММЫ НА ЯЗЫКЕ С++ СО СТРУКТУРАМИ ЦИКЛА

1. ЦЕЛЬ ЗАНЯТИЯ

Целью занятия является приобретение знаний и навыков составления программ с использованием операторов цикла for, do while и while.

2. ОПЕРАТОРЫ ЦИКЛА

2.1. Цикл с параметром for

Цикл с параметром имеет следующий формат:

```
for (инициализация; выражение; модификации) оператор;
```

Инициализация используется для объявления и присвоения начальных значений величинам, используемым в цикле. В этой части можно записать несколько операторов, разделенных запятой (операцией «последовательное выполнение»), например, так:

```
for (int i= 0, j = 2; …
int k, m;
for (k = 1, m = 0; …
```

Областью действия переменных, объявленных в части инициализации цикла, является цикл. Инициализация выполняется в начале исполнения цикла.

Выражение определяет условие выполнения цикла: если его результат, приведенный к типу bool, равен true, цикл выполняется. Цикл с параметром реализации как цикл с предусловием.

Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметров цикла. В части модификаций можно записать несколько операторов через запятую. Простой или составной оператор представляет собой тело цикла. Любая из частей оператора for может быть опущена (но точки с запятой надо оставить на своих местах!).

Приступая к решению задач этого раздела, следует помнить, что:

- инструкция for используется для организации циклов с фиксированным, известным во время разработки программы число повторений;
- количество повторений цикла определяется начальным значением переменной-счетчика и условием завершения цикла;
- переменная-счетчик должна быть целого (int) типа и может быть объявлена непосредственно в инструкции цикла.

Пример 1. Программа печатает таблицу значений функции $y=x^2+1$ во введенном диапазоне.

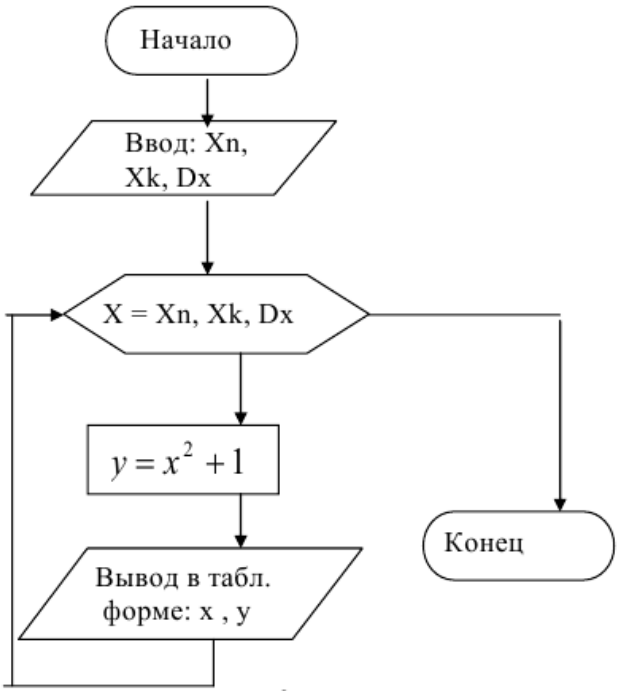
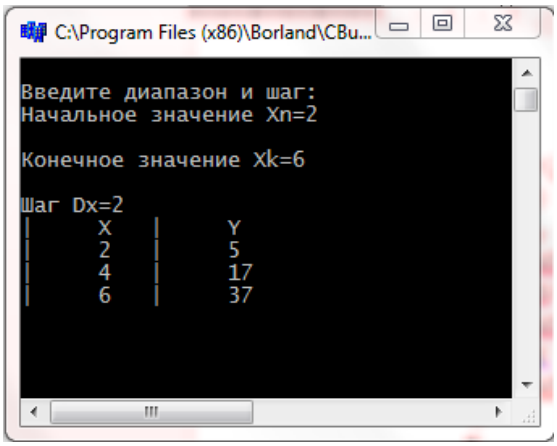


Рис. 1. Схема алгоритма вычисления функции $y=x^2+1$ с использованием оператора for

Unit1.cpp

```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#include <windows.h>  //Для отображения русских букв
#pragma hdrstop
//---------------------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
        SetConsoleOutputCP(1251); //Для отображения русских букв
        SetConsoleCP(1251);   //Для отображения русских букв
        int Xn, Xk, Dx,X;
        cout<<"\n"<<"Введите диапазон и шаг:";
        cout<<"\n"<<"Начальное значение Xn=";cin>>Xn;
        cout<<"\n"<<"Конечное значение Xk=";cin>>Xk;
        cout<<"\n"<<"Шаг Dx=";cin>>Dx;
        cout<<"|     X    |     Y      \n";
              for (int X=Xn;X<=Xk;X+=Dx) {
        cout<<"|     "<<X<<"    |     "<<X*X+1<<"\n";
        };
        getch();
}
```

26: 2 Modified Insert Code



2.2. Цикл с предусловием (while)

Цикл с предусловием реализует структурную схему, приведенную на рис. 2.а, и имеет вид:

while (выражение) оператор

Выражение определяет условие повторения тела цикла, представленного простым или составным оператором. Выполнение оператора начинается с вычисления выражения. Если оно истинно (не равно false), выполняется оператор цикла. Если при первой проверке выражение равно false, цикл не

Рис. 2. Структурные схемы операторов цикла: а – цикл с предусловием; б – цикл с постусловием

выполнится ни разу. Тип выражения должен быть арифметическим или приводимым к нему. Выражение вычисляется перед каждой итерацией цикла.

Приступая к решению задач этого раздела, следует помнить, что:

- число повторений инструкций цикла while определяется ходом выполнения программы;
- инструкции цикла while выполняются до тех пор, пока значение выражения, записанного после слова while, не станет равным нулю;
- после слова while надо записывать условие выполнения инструкций цикла;
- для завершения цикла while в теле цикла обязательно должны быть инструкции, выполнение которых влияет на условие завершения цикла;
- цикл while – это цикл с предусловием, т.е. возможна ситуация, при которой инструкции тела цикла ни разу не будут выполнены;
- цикл while, как правило, используется для организации приближенных вычислений, в задачах поиска и обработки данных, вводимых с клавиатуры или из файла.

Пример 2. Программа печатает таблицу значений функции $y=x^2+1$ во введенном диапазоне

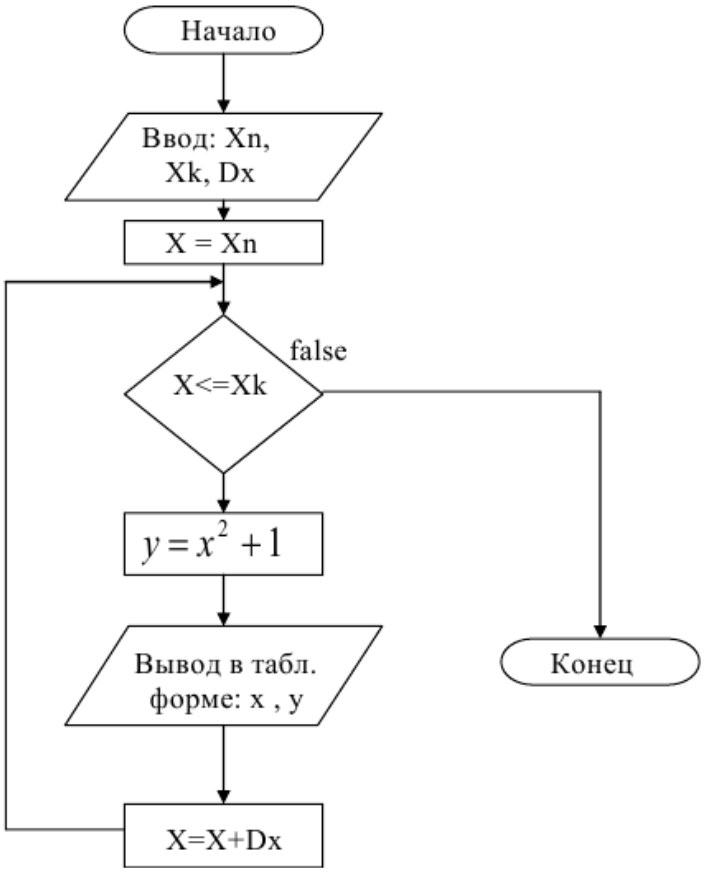
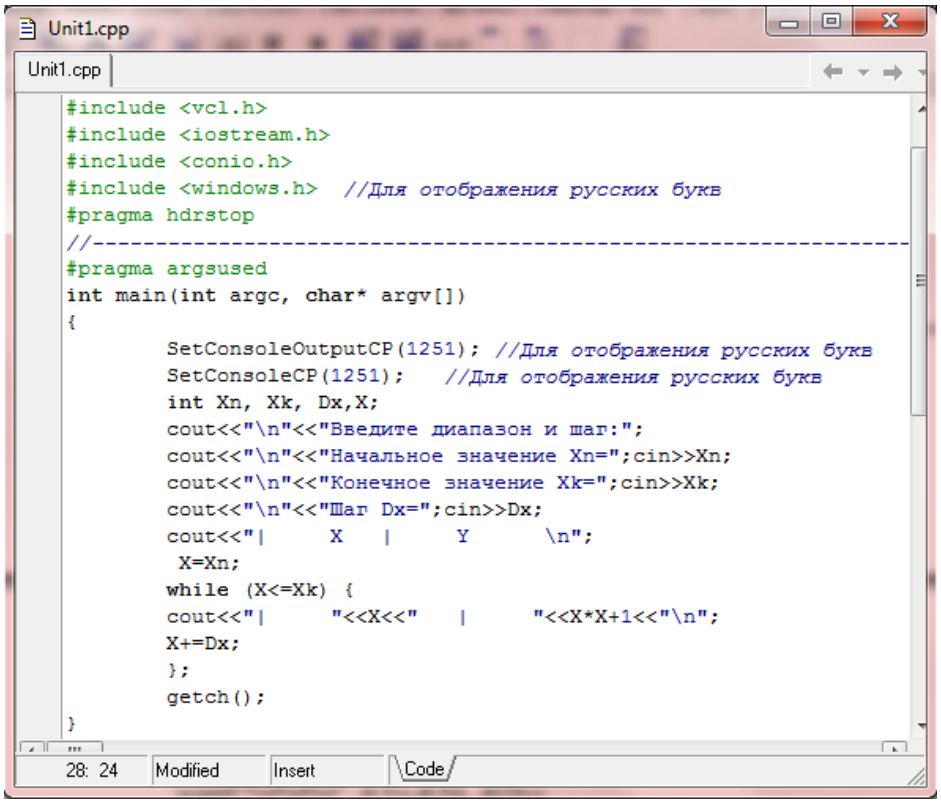


Рис. 3. Схема алгоритма вычисления функции $y=x^2+1$ с использованием цикла с предусловием



2.3. Цикл с постусловием (do while)

Цикл с постусловием реализует структурную схему, приведенную на рис. 2.б, и имеет вид:

do оператор while выражение;

Сначала выполняется простой или составной оператор, составляющий тело цикла, а затем вычисляется выражение. Если оно истинно (не равно false), тело цикла выполняется еще раз. Цикл выполняется до тех пор, пока выражение отлично от нуля, т.е. заключенное в нем условие цикла истинно true. Цикл завершается, когда выражение станет равным false или в теле цикла будет выполнен какой-либо оператор передачи управления. Тип выражения должен быть арифметическим или приводимым к нему.

Приступая к решению задач этого раздела, необходимо помнить, что:

- число повторений инструкций цикла do while определяется ходом выполнения программы;
- инструкции цикла do while выполняются до тех пор, пока значение выражения, записанного после слова while, не станет равно нулю;
- после слова while надо записывать условие выполнения инструкций цикла;
- для завершения цикла do while в теле цикла обязательно должны быть инструкции, выполнение которых влияет на условие завершения цикла;
- цикл do while – это цикл с постусловием, т.е. инструкции тела цикла будут выполнены хотя бы один раз;

- цикл do while, как правило, используется для организации приближенных вычислений, в задачах поиска и обработки данных, вводимых с клавиатуры или из файла.

Пример 3. Программа печатает таблицу значений функции $y=x^2+1$ во введенном диапазоне.

Фрагмент программного кода:

```
do {
cout<<"|      "<<X<<"    |      "<<X*X+1<<"\n";
X+=Dx;
}
while (X<=Xk);
```

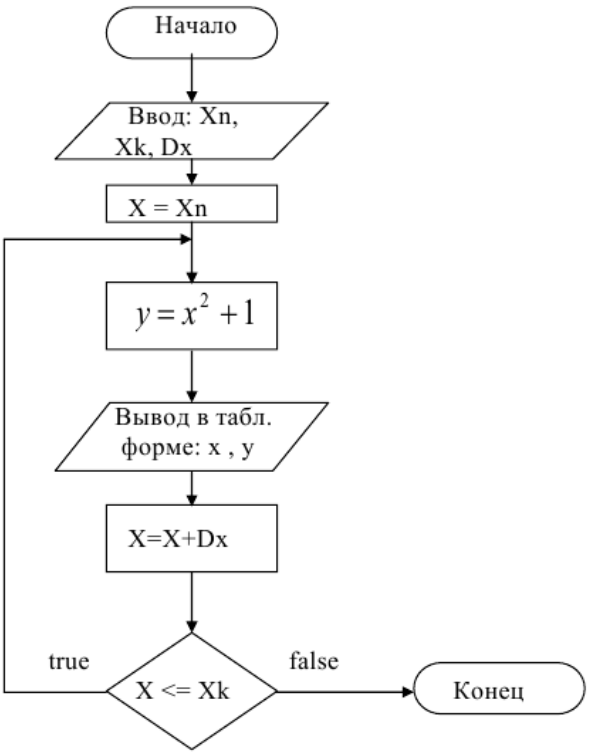
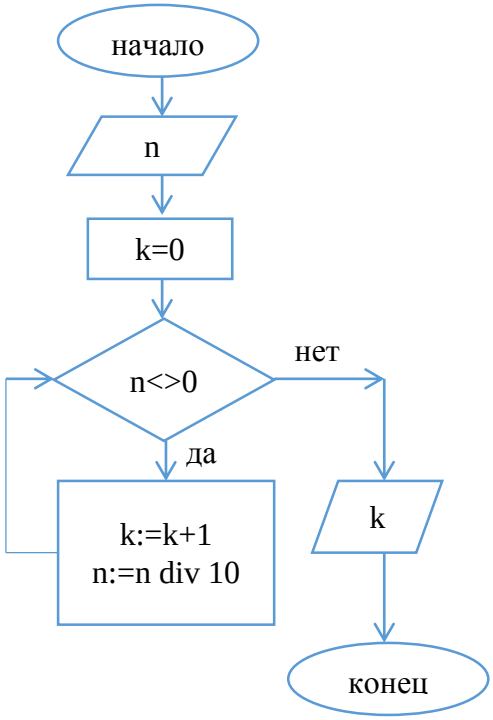


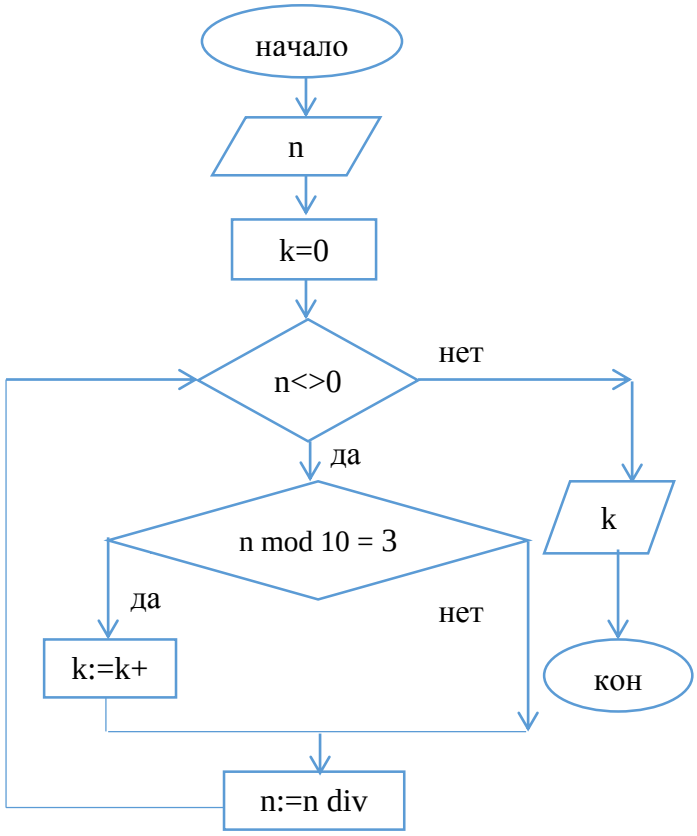
Рис. 4. Схема алгоритма вычисления функции $y=x^2+1$ с использованием цикла с постусловием

Пример 4. Определить количество цифр во введенном натуральном числе n.




```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#include <windows.h>
#include <math.h>
#pragma hdrstop
//---------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
        SetConsoleOutputCP(1251);
        SetConsoleCP(1251);
        int n, k=0;
        cout<<"\n"<<"Введите целое число:"; cin >> n;
        while (n!=0) { k++; n=floor(n/10);}
        cout<<"\n"<<"Количество цифр в числе "<<k;
        getch();
}
```

Пример 5. Дано натуральное число n. Определить количество цифр 3 в нем.




```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#include <windows.h>
#include <math.h>
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
int main(int argc, char* argv[])
{
        SetConsoleOutputCP(1251);
        SetConsoleCP(1251);
        int n, k=0;
        cout<<"\n"<<"Введите целое число:"; cin >> n;
        while (n!=0) {
        if (fmod(n,10)==3) k++;
        n=floor(n/10);
        }
        cout<<"\n"<<"Количество цифр 3 в числе "<<k;
        getch();
}
```

Пример 6. *Для заданного* ***x*** *и* ***n*** *вычислить*

$$S = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots + (-1)^n \frac{x^n}{n!}$$

Здесь n! = 1· 2· 3 ...· n (читается как "n-факториал").

```
int main(int argc, char* argv[])
{
        SetConsoleOutputCP(1251);
        SetConsoleCP(1251);
        int n, x;
        double S=1, p=1;
        cout<<"\n"<<"Введите число n="; cin >> n;
        cout<<"\n"<<"Введите число x="; cin >> x;
        for (int i=1; i<=n;i++) {
        p=-p*x/i;
        S=S+p;
        }
        cout<<"\n"<<"S="<<S;
        getch();
}
```

Пример 7. Определить сумму ряда $S = 1 + \frac{1}{3^2} + \frac{1}{5^2} + \cdots + \frac{1}{(2n+1)^2}$, пока очередной член ряда не станет меньше 10^{-8}.

```
int main(int argc, char* argv[])
{
        SetConsoleOutputCP(1251);
        SetConsoleCP(1251);
        int n=1;
        double S=0, a,eps;  //eps - точность вычислений
        eps=1e-8;
        do {
        a=1/pow((2*n+1),2); // текущий член ряда
        n++;
        S=S+a;
        }
        while(a>eps);
        cout<<"\n"<<"S="<<S;
        cout<<"\n"<<"n="<<n; //количество итераций
        getch();
}
```

(ответ: S=0.233651, n=5001)

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Циклический процесс. Вычисление значений функции

Разработать программу на языке С++. Для решения задачи использовать операторы for, while, do. Варианты заданий:

1. Вывести на экран таблицу значений функции синус в диапазоне от $-2\cdot\pi$ до $2\cdot\pi$ с шагом $\pi/8$.
2. Вывести на экран таблицу квадратов первых десяти целых положительных чисел.
3. Вывести на экран таблицу значений функции косинус в диапазоне от $-2\cdot\pi$ до $2\cdot\pi$ с шагом $\pi/8$.
4. Вывести на экран таблицу кубов первых десяти целых положительных чисел.
5. Вывести на экран таблицу значений квадратов синусов в диапазоне от $-\pi$ до π с шагом $\pi/12$.
6. Вывести на экран таблицу значений квадратов косинусов в диапазоне от 0 до 2π с шагом $\pi/10$.
7. Вывести на экран таблицу квадратов первых десяти целых четных положительных чисел.
8. Вывести на экран таблицу квадратов первых десяти целых нечетных положительных чисел.
9. Вывести на экран таблицу значений удвоенных синусов в диапазоне от $-a$ до a с шагом h . Значения a и h вводятся с клавиатуры.
10. Вывести на экран таблицу значений удвоенных косинусов в диапазоне от a до b с шагом h . Значения a , b и h вводятся с клавиатуры.
11. Вывести на экран таблицу кубов первых десяти целых нечетных положительных чисел.
12. Вывести на экран таблицу кубов первых десяти целых четных положительных чисел.
13. Вывести на экран таблицу значений функции $y=e^{2x}$ в диапазоне от $-a$ до a с шагом h . Значения a и h вводятся с клавиатуры.
14. Вывести на экран таблицу значений функции $y=5\cdot e^{-3x}$ в диапазоне от a до b с шагом h . Значения a , b и h вводятся с клавиатуры.
15. Вывести на экран таблицу квадратов первых десяти целых отрицательных чисел.
16. Вывести на экран таблицу кубов первых десяти целых отрицательных чисел.
17. Вывести на экран таблицу квадратных корней первых десяти целых положительных чисел.
18. Вывести на экран таблицу кубических корней первых десяти целых положительных чисел.

19. Вывести на экран таблицу значений функции $y=2\cdot x^2+3\cdot x-1$ в диапазоне от $-a$ до a с шагом h. Значения a и h вводятся с клавиатуры.
20. Вывести на экран таблицу значений функции $y=5.4\cdot x^3-2.8\cdot x^2-x+1.6$ в диапазоне от a до b с шагом h. Значения a, b и h вводятся с клавиатуры.
21. Вывести на экран таблицу квадратных корней первых десяти целых положительных четных чисел.
22. Вывести на экран таблицу квадратных корней первых десяти целых положительных нечетных чисел.
23. Вывести на экран таблицу значений функции $y=-1.8\cdot x^3-e^{2x}+1/6$ в диапазоне от -3 до 4 с шагом $1/2$.
24. Вывести на экран таблицу значений функции $y=-1.3\cdot x^2-e^x/4$ в диапазоне от -2 до 2 с шагом $1/4$.
25. Вывести на экран таблицу степеней двойки в диапазоне от 0 до 10 с шагом 1.

2. Циклический процесс. Последовательности натуральных чисел

Разработать программу на языке C++ для следующих заданий:
1. Дано целое положительное число N. Вычислить сумму натуральных нечетных чисел не превышающих это число.
2. Дано целое положительное число N. Вычислить произведение натуральных четных чисел не превышающих это число.
3. Дано целое положительное число N. Вычислить количество натуральных чисел кратных трем и не превышающих число N.
4. Задано целое положительное число n. Определить значение выражения:

$$P = \frac{n!}{\sum_{i=1}^{n} i}.$$

5. Вычислить количество натуральных двузначных четных чисел не делящихся на 10.
6. Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=1}^{n} i^2}{n!}$$

7. Вычислить сумму натуральных удвоенных чисел не превышающих 25.
8. Задано целое положительное число n. Определить значение выражения:
$P = \frac{\sum_{i=3}^{n} i-2}{(n+1)!}$.
9. Дано целое положительное число N. Вычислить сумму квадратов натуральных четных чисел не превышающих это число.
10. Дано целое положительное число N. Вычислить количество натуральных чисел кратных пяти и не превышающих число N.
11. Определить значение выражения:

$$P = \frac{\sum_{i=0}^{5} 3^i}{5!}$$

.
12. Дано целое положительное число N. Вычислить сумму удвоенных натуральных нечетных чисел не превышающих это число.

13. Задано целое положительное число n. Определить значение выражения:

$$P = \sum_{i=2}^{n} i^2 - 1$$

14. Найти сумму нечетных степеней двойки. Значение степени изменяется от 1 до 9.
15. Задано целое положительное число n. Определить значение выражения:

$$P = \frac{1}{3} \cdot \sum_{i=1}^{n} 2 \cdot i^2 - i + 1.$$

16. Дано целое положительное число N. Вычислить произведение натуральных чисел кратных трем и не превышающих число N.
17. Задано целое положительное число n. Определить значение выражения:

$$P = \sum_{i=3}^{n+2} 2 \cdot i - 4.$$

18. Вычислить сумму натуральных трехзначных чисел кратных пяти и не делящихся на десять.
19. Определить значение выражения: $P = \sum_{i=0}^{10} 2^i$
20. Вычислить количество натуральных двузначных нечетных чисел не делящихся на 5.
21. Задано целое положительное число n. Определить значение выражения:
22. Задано целое положительное число n. Определить значение выражения:

$$P = \frac{\sum_{i=5}^{15} i}{(2 \cdot n+1)!}.$$

23. Найти произведение четных степеней двойки. Значение степени изменяется от 0 до 8.
24. Вычислить произведение натуральных чисел не превышающих 15.
25. Вычислить произведение натуральных двузначных чисел кратных трем и не делящихся на 10.

3. Циклический процесс. Последовательности произвольных чисел

Разработать программу на языке C++ для следующих заданий:
1. Вводится последовательность ненулевых чисел, 0 – конец последовательности. Определить сумму положительных элементов последовательности.
2. Вычислить сумму отрицательных элементов последовательности из N произвольных чисел.
3. Вводится последовательность ненулевых чисел, 0 – конец последовательности. Определить сколько раз последовательность поменяет знак.
4. В последовательности из N произвольных чисел подсчитать количество нулей.
5. Вводится последовательность ненулевых чисел, 0 – конец последовательности. Определить наибольшее число в последовательности.
6. Вводится последовательность из N произвольных чисел найти наименьшее число в последовательности.
7. Вводится последовательность ненулевых чисел, 0 – конец последовательности. Определить среднее значение элементов последовательности.

8. Вводится последовательность из N произвольных чисел найти среднее значение положительных элементов последовательности.
9. Вводится последовательность ненулевых чисел, 0 – конец последовательности. Подсчитать процент положительных и отрицательных чисел.
10. Вводится последовательность из N произвольных чисел. Определить процент положительных, отрицательных и нулевых элементов.
11. Вводится последовательность из N произвольных чисел. Вычислить разность между наименьшим и наибольшим значениями последовательности.
12. Вводится последовательность из N положительных целых чисел. Найти наименьшее число среди четных элементов последовательности.
13. Вводится последовательность из N положительных целых чисел. Определить является ли эта последовательность знакочередующейся.
14. Определить является ли последовательность из N произвольных чисел строго возрастающей (то есть каждый следующий элемент больше предыдущего).
15. Вводится последовательность произвольных чисел, 0 – конец последовательности. Определить является ли эта последовательность строго убывающей (то есть каждый следующий элемент меньше предыдущего).
16. Вводится последовательность ненулевых целых чисел, 0 – конец последовательности. Определить среднее значение четных элементов последовательности.
17. Вводится последовательность из N произвольных чисел найти среднее значение отрицательных элементов последовательности.
18. В последовательности из N целых чисел подсчитать четных и нечетных чисел.
19. Вводится последовательность целых чисел, 0 – конец последовательности. Определить процент четных и нечетных чисел в последовательности.
20. Вводится последовательность из N целых чисел. Определить содержит ли последовательность хотя бы два соседних одинаковых числа.
21. Вводится последовательность целых чисел, 0 – конец последовательности. Определить наибольшее число среди нечетных элементов последовательности.
22. Вводится последовательность произвольных чисел, 0 – конец последовательности. Определить сумму и количество чисел в последовательности.
23. Вводится последовательность из N произвольных чисел. Найти сумму положительных и сумму отрицательных элементов последовательности.
24. Вводится последовательность произвольных чисел, 0 – конец последовательности. Определить отношение минимального и максимального элементов друг к другу.
25. Вводится последовательность из N целых чисел. Определить количество одинаковых рядом стоящих чисел.

4. Циклический процесс. Работа с цифрами в числе

Разработать программу на языке C++ для следующих заданий:

1. Определить является ли целое положительное число совершенным. Совершенное число равно сумме всех своих делителей, не превосходящих это число. Например, 6=1+2+3 или 28=1+2+4+7+14.
2. Проверить является ли пара целых положительных чисел дружественными. Два различных натуральных числа являются дружественными, если сумма всех делителей первого числа (кроме самого числа) равна второму числу. Например, 220 и 284, 1184 и 1210, 2620 и 2924, 5020 и 5564.
3. Определить является ли целое положительное число недостаточным. Недостаточное число всегда больше суммы всех своих делителей за исключением самого числа.
4. Вводится целое положительное число. Определить количество четных и нечетных цифр в числе.
5. Вводится целое положительное число. Найти число, которое равно сумме кубов цифр исходного числа.
6. Задача о «счастливом» билете. Вводится целое положительное шестизначное число. Определить совпадает ли сумма первых трех цифр с суммой трех последних.
7. Задача о «встречном» билете. Вводится целое положительное шестизначное число. Убедиться, что разница между суммой первых трех цифр и суммой последних трех цифр равна единице.
8. Задано целое положительное число. Определить количество его четных и нечетных делителей.
9. Проверить является ли два целых положительных числа взаимно простыми. Два различных натуральных числа являются взаимно простыми, если их наибольший общий делитель равен единице.
10. Определить является ли целое положительное число составным. Составное число имеет более двух делителей, то есть не является простым.
11. Вводится целое положительное число. Найти наименьшую цифру числа.
12. Задано целое положительное число. Определить является ли оно числом Армстронга. Число Армстронга — натуральное число, которое равно сумме своих цифр, возведенных в степень, равную количеству его цифр. Например, десятичное число 153 — число Армстронга, потому что: $1^3+3^3+5^3=1+27+125=153$.
13. Вводится целое положительное число. Найти произведение всех ненулевых цифр числа.
14. Вводится целое положительное число. Найти наибольшую цифру числа.
15. Вводится целое положительное число. Определить позицию наибольшей цифры в числе.
16. Вводится целое положительное число. Найти число, которое равно сумме удвоенных цифр исходного числа.
17. Вводится целое положительное число. Найти число, которое равно сумме квадратов цифр исходного числа.

18. Задано целое положительное число. Определить сумму его делителей.
19. Вводится целое положительное число. Определить позицию наименьшей цифры в числе.
20. Проверить, что два целых положительных числа не являются взаимно простыми. Различные натуральные числа не являются взаимно простыми, если их наибольший общий делитель отличен от единицы.
21. Убедиться, что заданное целое положительное число не является палиндромом. Числа палиндромы симметричны относительно своей середины, например, 12021 или 454.
22. Убедиться, что заданное целое положительное число не является совершенным. Совершенное число равно сумме всех своих делителей, не превосходящих это число. Например, 6=1+2+3 или 28=1+2+4+7+14.
23. Проверить, что два целых положительных числа не являются дружественными. Два различных натуральных числа являются дружественными, если сумма всех делителей первого числа (кроме самого числа) равна второму числу. Например, 220 и 284, 1184 и 1210, 2620 и 2924, 5020 и 5564.
24. Вводится целое положительное число. Найти число, которое равно сумме утроенных цифр исходного числа.
25. Вводятся два целых положительных числа. Найти сумму их цифр.

5. Вложенные циклы

Разработать программу на языке C++ для следующих заданий:
1. Дано натуральное число P. Вывести на печать все простые числа не превосходящие P.
2. Дано натуральное число P. Вывести на печать все совершенные числа не превосходящие P.
3. Вводится последовательность положительных целых чисел, 0 – конец последовательности. Определить количество совершенных чисел.
4. Вводится последовательность положительных целых чисел, 0 – конец последовательности. Определить количество простых чисел.
5. Вводится последовательность из N положительных целых чисел. Для каждого элемента последовательности вычислить факториал.
6. Вводится последовательность из N положительных целых чисел. Вывести на печать все числа — палиндромы. Если таких чисел нет, выдать соответствующее сообщение.
7. Вводится последовательность из N положительных целых чисел. Определить разрядность каждого числа.
8. Вводится последовательность из N положительных целых чисел. Вывести на печать количество делителей каждого числа.
9. Вводится последовательность положительных целых чисел, 0 – конец последовательности. Определить сумму цифр каждого элемента последовательности.

10. Дано K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Для каждого набора вывести количество его элементов. Вычислить общее количество элементов.
11. Дано K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Для каждого набора вычислить среднее арифметическое его элементов.
12. Даны K наборов целых чисел по N элементов в каждом наборе. Для каждого набора найти наибольшее значение его элементов.
13. Даны K наборов целых чисел по N элементов в каждом наборе. Определить есть ли среди наборов данных знакочередующиеся последовательности.
14. Даны K наборов целых чисел по N элементов в каждом наборе. Определить есть ли среди наборов данных строго возрастающие последовательности.
15. Дано K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Для каждого набора найти наименьшее значение его элементов.
16. Даны K наборов целых чисел по N элементов в каждом наборе. Для каждого набора вычислить произведение не нулевых элементов.
17. Даны K наборов целых чисел по N элементов в каждом наборе. Найти наибольшее число для всех наборов.
18. Дано K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Вычислить среднее арифметическое всех элементов во всех наборах.
19. Дано K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Найти количество возрастающих наборов.
20. Дано K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Найти количество убывающих наборов.
21. Дано K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Найти количество наборов не являющихся знакочередующимися.
22. Дано K наборов ненулевых целых чисел. Признаком завершения каждого набора является число 0. Найти количество наборов элементы которых не возрастают и не убывают.
23. Даны целые положительные числа N и M (N < M). Вывести все целые числа от N до M включительно; при этом каждое число должно выводиться столько раз, каково его значение (например, число 5 выводится 5 раза).
24. Дано целое число N (> 0). Найти сумму 1! + 2! + 3! + … + N!
25. Даны целые числа N и M (N < M). Вывести все целые числа от N до M включительно; при этом число N должно выводиться 1 раз, число N + 1 должно выводиться 2 раза и т. д.

ТЕМА 5. ОБРАБОТКА МАССИВОВ В С++ С++ СО СТРУКТУРАМИ ЦИКЛА

1. ЦЕЛЬ ЗАНЯТИЯ

Целью занятия является приобретение знаний и навыков составления программ с использованием одномерных и многомерных массивов.

2. МАССИВЫ

Если с группой величин одинакового типа требуется выполнять однообразные действия, им дают одно имя, а различают по порядковому номеру. Это позволяет компактно записывать множество операций с помощью циклов. Конечная именованная последовательность однотипных величин называется массивом.

2.1. Одномерные массивы

При работе с массивами необходимо научиться решать три основные задачи:

- Как правильно выделять память нужного размера под массив?
- Как правильно записывать данные в нужную ячейку?
- Как правильно научиться читать данные из ячейки массива?

Описание массива в программе отличается от описания простой переменной наличием после имени квадратных скобок, в которых задается количество элементов массива, т.е. его размерность: Формат описания массива следующий:

```
тип Имя_массива [размер];
```

Здесь тип определяет тип данных для каждого элемента в массиве, а размер указывает количество элементов в массиве. Например,

```
int  x[10];   // описание массива из 10 целочисленных элементов
float A[8];    // описание массива из 8 вещественных чисел
char asd[20]; //описание массива из 20 символов
```

Элементы массива нумеруются с нуля, например, согласно описанию объявлен массив с именем x, содержащий 5 элементов целого типа:

```
int  x[5].
```

Элементы массива еще называют индексированными именами. Нижнее значение индекса равно 0:

```
x[0], x[1], x[2]. x[3], x[4].
```

Массивы можно инициализировать с помощью списка инициализаторов, заключенного в фигурные скобки, как, например, показано ниже:

```
int A[5] = {5, 3, 7, 2, 9};
```

Если при объявлении массива производится инициализация значений элементов, то размер массива может не указываться:

```
int A[ ] = {5, 3, 7, 2, 9};
```

В этом случае создается массив из пяти элементов со следующими значениями:

```
A[0] = 5, A[1] = 3, A[2] = 7, A[3] = 2, A[4] = 9.
```

Можно задать массив еще и таким образом:

```
int M[6] = {5, 3, 2};
```

В этом случае будет создан массив из шести элементов. Первые три элемента получат инициализированные значения. Значения остальных будут равны нулю:

```
M[0] = 5, M[1] = 3, M[2] = 2, M[3] = 0, M[4] = 0, M[5] = 0;
```

Размерность массива вместе с типом его элементов определяет объем памяти, необходимы для размещения массива, которое определяется на этапе компиляции, поэтому размерность может быть задана только целой положительной константой или константным выражением. Если при описании массива не указана размерность, должен присутствовать инициализатор, в этом случае компилятор выделит память по количеству инициализирующих значений.

2.2. Многомерные массивы

Многомерные массивы задаются указанием каждого измерения в квадратных скобках, например, оператор

```
int matr [4] [3];
```

задает описание двумерного массива из 4 строк и 3 столбцов. В памяти такой массив располагается построчно. Многомерные массивы размещаются так, что при переходе к следующему элементу быстрее всего изменяется последний индекс. Для доступа к элементу многомерного массива указывают все его индексы, например, matr [i] [j].

При инициализации многомерного массива он представляется либо как массив из массивов, при этом каждый массив заключается в свои фигурные скобки (в этом случае левую размерность при описании можно не указывать):

```
int mass [ ][2] = { {1, 1}, {0, 2}, {1, 0} };
```

либо задается общий список элементов в том порядке, в котором элементы располагаются в памяти:

```
int mass [3][2] = {1, 1, 0, 2, 1, 0};
```

Итак, приступая к решению задач данного раздела, следует помнить, что:

- массив – структура, представляющая собой набор, совокупность элементов одного типа;
- в инструкции объявления массива указывается количество элементов массива;
- элементы массива нумеруются с нуля;
- доступ к элементу массива осуществляется путем указания индекса (номера) элемента. В качестве индекса можно использовать выражение целого типа –константу или переменную. Индекс может меняться от 0 до n – 1, где n –количество элементов массива;
- в инструкции объявления массива удобно использовать именованную константу, объявленную в директиве #define, например:

```
#include <stdio.h>
#define  size 5    //размер массива
void main ()
{        int a[size];              // массив
…… }
```

- для ввода, вывода и обработки массивов удобно использовать инструкции циклов (for, while);
- типичной ошибкой при использовании массивов является обращение к несуществующему элементу, т.е. выход индекса за допустимое значение.

Пример 1. Подсчет суммы элементов массива.
Дан массив `m[n] = {3, 5, 7, 9, 4, 4, 10}`, состоящий из 10 элементов (n = 10).
Необходимо подсчитать сумму элементов массива.

Программа к примеру 1:

```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop

//---------------------------------------

#pragma argsused
int main(int argc, char* argv[])
{
      const int n=10;
      int i,sum;
      int m[n]={3,5,7,9,4,4,10};
      for (i=0,sum=0;i<n;i++) sum+=m[i];
      cout<<"Summa:"<<sum<<endl;
      getch();
}
```

Измените код таким образом, чтобы пользователь мог ввести элементы массива с клавиатуры.

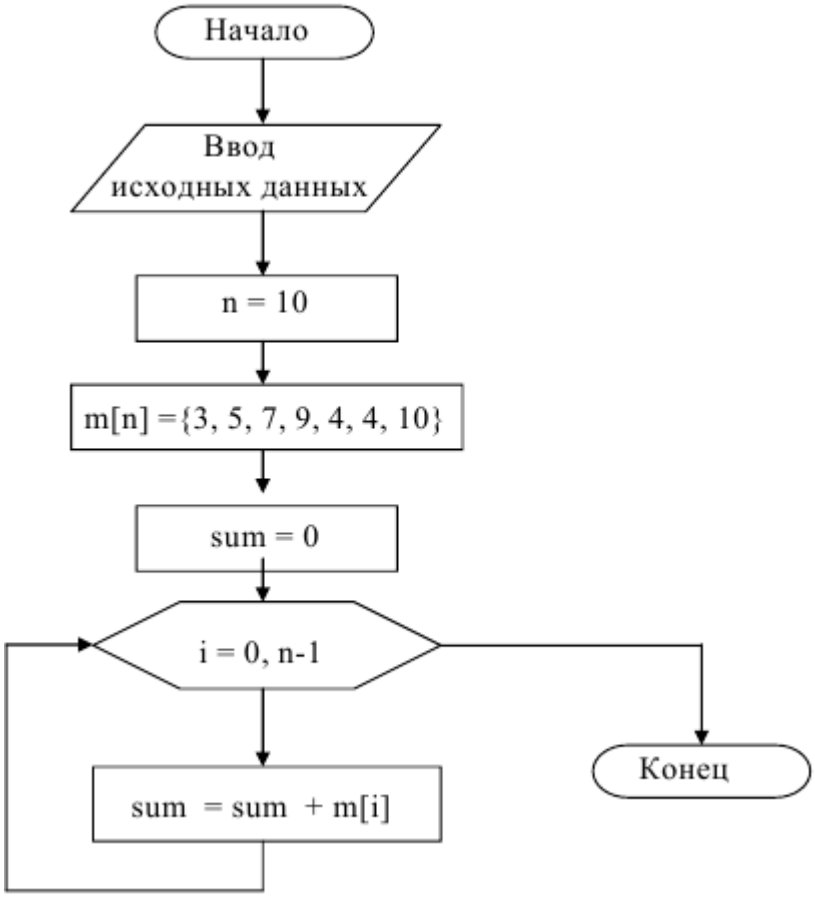


Рис. 1. Структурная схема алгоритма вычисления суммы элементов

Пример 2. Произвести сортировку целочисленного массива методом выбора. Алгоритм состоит в том, что выбирается наименьший элемент массива и меняется местами с первым элементом, затем рассматриваются элементы, начиная со второго, и наименьший из них меняется местами со вторым элементом, и так далее n-1 раз (при последнем проходе цикла при необходимости меняются местами предпоследний и последний элементы массива). Процесс обмена элементов массива с номерами i и min через буферную переменную a на i-том проходе цикла проиллюстрирован на рис. 2. Цифры около стрелок обозначают порядок действий.

Структурная схема алгоритма сортировки элементов массива приведена на рис. 3.

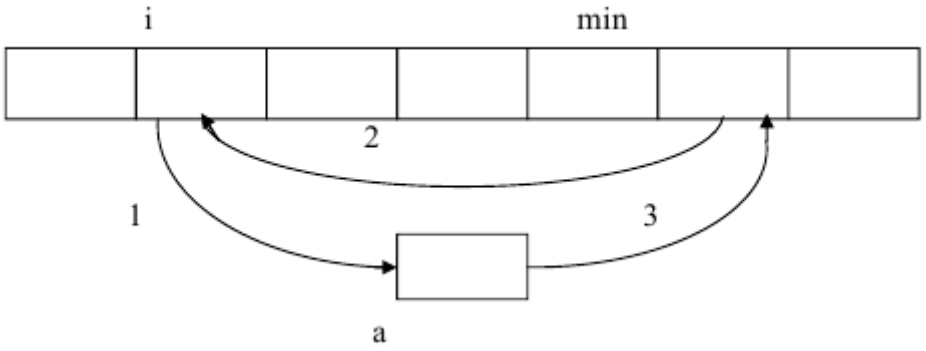


Рис. 2. Обмен значений двух переменных

Программа к примеру 2 на языке C++.

```
int main(int argc, char* argv[])
{          const int n=5;
      int i;
      int b[n];
      for (i=0;i<n;i++) {cout<<"b["<<i+1<<"]="; cin>>b[i]; }
      for (i=0;i<n-1;i++)
        {int min=i;
                for (int j=i+1;j<n;j++)
                        if (b[j]<b[min]) min=j;
                        int a=b[i];
                        b[i]=b[min];
                        b[min]=a;}
                for (i=0;i<n;i++)
                        {cout<<"b["<<i+1<<"]="<<b[i]<<"    ";}
                cout<<"\n";

      getch();}
```

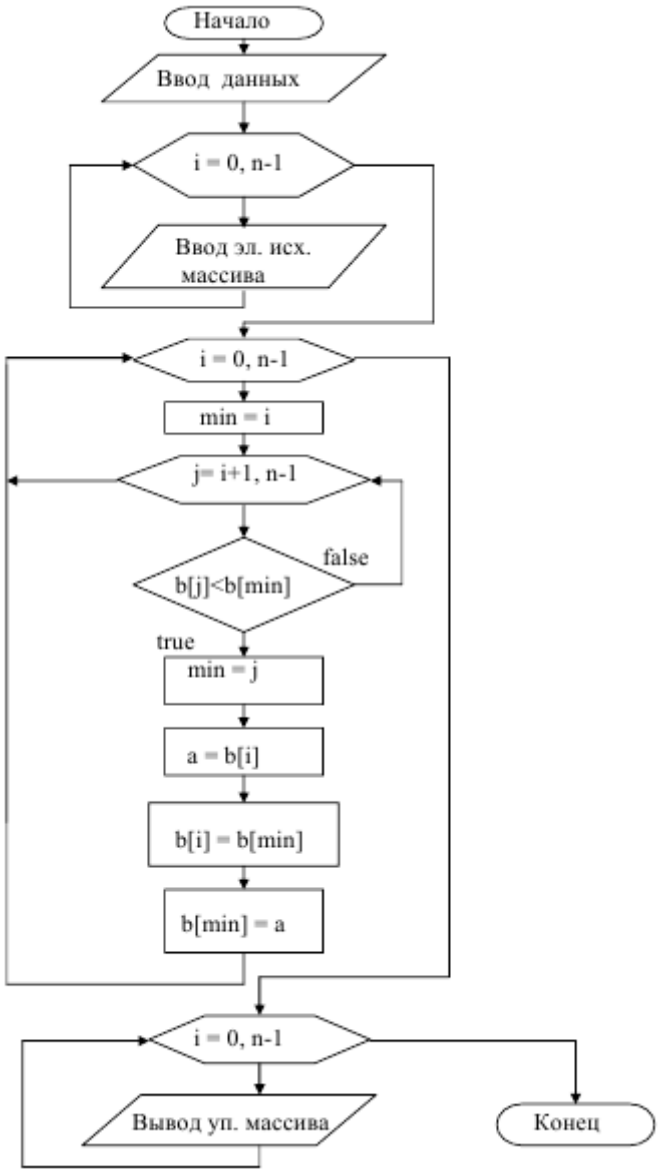


Рис. 3. Структурная схема алгоритма сортировки элементов массив

Пример 3. Программа вводит с клавиатуры двухмерный массив

```
#include <iostream.h>
void main()
{  const int n=2, k=3;
    int M[n][k];
    int i,j;
  for (i=0;i<n;i++)
    for(j=0;j<k;j++)
  {cout<<"M["<<i+1<<"]["<<j+1<<"]=";
    cin>>M[i][j];}}
```

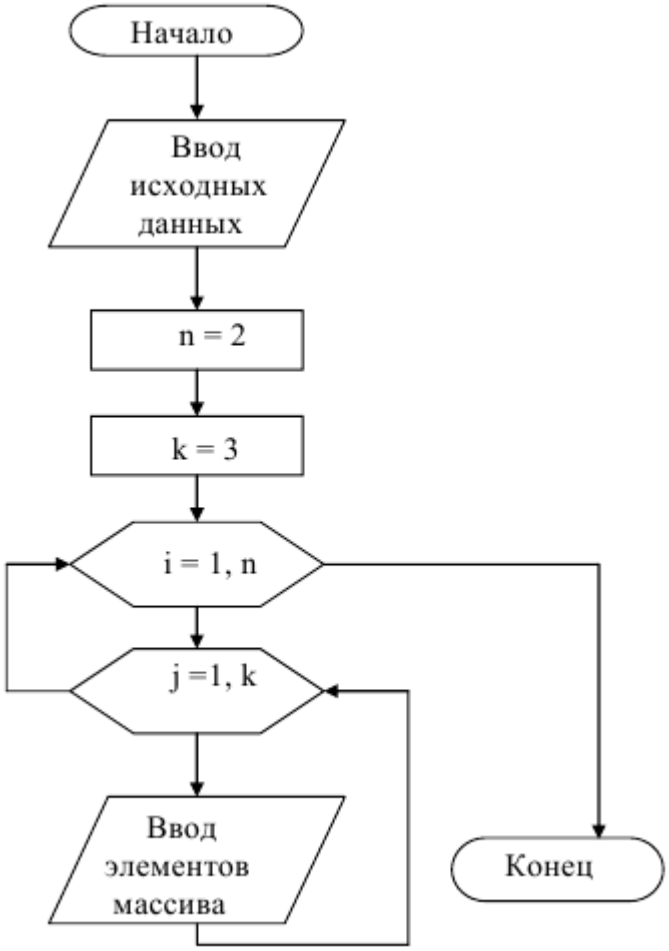


Рис. 4. Структурная схема алгоритма ввода элементов двухмерного массива

2.3. Строки

Строка представляет собой массив символов, заканчивающийся нуль-символом. Нуль-символ – это символ с кодом, равным 0, что записывается в виде управляющей последовательности '\0'. По положению нуль-символа определяется фактическая длина строки. Строка описывается как символьный массив. Например:

```
char STR[20];
```

Одновременно с описанием строка может инициализироваться. Возможны два способа инициализации строки – с помощью строковой константы и в виде списка символов:

```
char S[10]= "строка";
char S[] = "строка";
```

```
char S[10] = {'с', 'т', 'р', 'о', 'к', 'а','\ 0'};
```

По результату первого описания под строку S будет выделено 10 байт памяти, из них первые 7 получат значения по инициализации (седьмой – нулевой символ). Второе описание сформирует строку из семи символов. Третье описание по результату равнозначно первому. Конечно, можно определить символьный массив и так:

```
char S[10] = {'с', 'т', 'р', 'о', 'к', 'а'};
```

т.е. без нулевого символа в конце. Но это приведет к проблемам с обработкой такой строки, т.к. будет отсутствовать ориентир на его окончание. Отдельные символы строки идентифицируются индексированными именами. Например, в описанной выше строке `S[0] = 'с', S[5] = 'а'.`

Обработка строк обычно связана с перебором всех символов от начала до конца. Признаком конца такого перебора является обнаружение нулевого символа.

Пример 4. В следующей программе производится последовательная замена всех символов строки на звездочки и подсчет длины строки.

```
#include <stdio.h>
#include <conio.h>
void main()
{  char S[]="abcd";
     int i=0;
  puts (S);
  while (S[i])
  {S[i++]='*'; puts (S);}
  printf("\nДлина строки =%i\n", i);}
```

Для вывода строки на экран в стандартной библиотеке stdio имеется функция puts ().
Аргументом этой функции указывается имя строки. В этой же библиотеке есть функция ввода строки с клавиатуры с именем gets(). В качестве аргумента указывается имя строки, в которую производится ввод.

Среди стандартных библиотек языка существует библиотека функций для обработки строк. Ее заголовочный файл – string.h.

Приступая к решению задач этого раздела, следует помнить, что:

- каждому символу соответствует число – код символа;
- в С++ строка – это массив символов;
- последним символом строки обязательно должен быть нуль – символ, код которого равен 0), и который в тексте программы изображается так: '\ 0';

- если вводимая во время работы программы строка содержит пробелы, то функция scanf вводит только часть строки до первого пробела, а функция gets – всю строку, в том числе и соответствующий клавише <Enter> символ '\ n '.

Пример 5. Ввод строки с клавиатуры и вывод строки на экран.

Программа к примеру 3 на языке С++.

```
#include<iostream.h>
#include<string.h>
void main()
{char str[10];
cout<<"Stroka:"<<endl;cin>>str;
cout<<str<<endl;}
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Написать программу, которая вводит с клавиатуры одномерный массив из 5 целых чисел, после чего выводит количество ненулевых элементов. Использовать средства ввода/вывода языка С++.
2. Написать программу, которая выводит минимальный элемент введенного с клавиатуры одномерного массива целых чисел. Использовать средства ввода/вывода языка С++.
3. Написать программу, которая выводит максимальный элемент введенного с клавиатуры одномерного массива целых чисел. Использовать средства ввода/вывода языка С++.
4. Написать программу, которая вычисляет среднюю за неделю температуру воздуха. Исходные данные вводятся с клавиатуры во время работы программы. Использовать средства ввода/вывода языка С++.
5. Написать программу, которая вычисляет среднее арифметическое ненулевых элементов введенного с клавиатуры одномерного массива целых чисел. Использовать средства ввода/вывода языка С++.
6. Написать программу, которая проверяет, есть ли во введенном с клавиатуры одномерном массиве элементы с одинаковыми значениями. Использовать средства ввода/вывода языка С++.
7. Написать программу, которая определяет количество студентов в группе, чей рост превышает средний. Использовать средства ввода/вывода языка С++.

8. Написать программу, которая вводит по строкам с клавиатуры двумерный массив и вычисляет сумму его элементов по столбцам. Использовать средства ввода/вывода языка C++.
9. Написать программу, которая вводит по строкам с клавиатуры двумерный массив и вычисляет сумму его элементов по строкам. Использовать средства ввода/вывода языка C++.
10. Написать программу, которая определяет в целочисленной двухмерной матрице номер строки, которая содержит наибольшее количество элементов, равных 0. Использовать средства ввода/вывода языка C++.
11. Написать программу, которая определяет строку квадратной матрицы, сумма элементов которой максимальна. Использовать средства ввода/вывода языка C++.
12. Написать программу, которая запрашивает имя пользователя и здоровается с ним. Использовать средства ввода/вывода языка C++.
13. Написать программу, которая вычисляет длину введенной с клавиатуры строки. Использовать средства ввода/вывода языка C++.
14. Написать программу, которая запрашивает название любимого фильма и выводит его на экран. Использовать средства ввода/вывода языка C++.

ТЕМА 6. ПРОГРАММЫ НА ЯЗЫКЕ С++ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ

1. ЦЕЛЬ ЗАНЯТИЯ

Целью занятия является приобретение знаний и навыков составления программ с использованием функций.

2. ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ ПРИ ПРОГРАММИРОВАНИИ НА С++

В практике программирования часто складываются ситуации, когда одну и ту же группу операторов, реализующих определенную цель, требуется повторить без изменений в нескольких местах программы. Для избавления от столь нерациональной траты времени была предложена концепция подпрограммы.

Подпрограмма – именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке С++ подпрограммы реализованы в виде функций.

2.1 Общие сведения о функциях. Локальные и глобальные переменные

Функция – это поименованный набор описаний и операторов, выполняющих определенную задачу. Функция может принимать параметры и возвращать значение. Информация, передаваемая в функцию для обработки, называется параметром, а результат вычислений функции ее значением. Обращение к функции называют вызовом.

Как известно, любая программа на С++ состоит из одной или нескольких функций. При запуске программы первой выполняется функция main. Если среди операторов функции main встречается вызов функции, то управление передается операторам функции. Когда все операторы функции будут выполнены, управление возвращается оператору, следующему за вызовом функции.

Перед вызовом функция должны быть обязательно описана. Описание функции состоит из заголовка и тела функции:

```
тип имя_функции(список_переменных)
{
тело_функции
}
```

Заголовок функции содержит:

• тип, возвращаемого функцией значения, он может быть любым; если функция не возвращает значения, указывают тип void;

• имя_функции, с которым она будет вызываться;

• список_переменных – перечень передаваемых в функцию величин (аргументов), которые отделяются друг от друга запятыми; для каждой переменной из списка указывается тип и имя; если функция не имеет аргументов, то в скобках указывают либо тип void, либо ничего.

Тело функции представляет собой последовательность описаний и операторов, заключенных в фигурные скобки.

В общем виде структура программы на C++ может иметь вид:

```
директивы компилятора
тип имя_1(список_переменных)
{
тело_функции_1;
}
тип имя_2(список_переменных)
{
тело_функции_2;
}
...
тип имя_n(список_переменных)
{
тело_функции_n;
}
int main(список_переменных)
{
//Тело функции может содержать операторы вызова
//функций имя_1, имя_2, ..., имя_n
тело_основной_функции;
}
```

Однако допустима и другая форма записи программного кода:

```
директивы компилятора
тип имя_1(список_переменных);
тип имя_2(список_переменных);
...
тип имя_n(список_переменных);
int main(Список_переменных)
{
//Тело функции может содержать операторы вызова
//функций имя_1, имя_2, ..., имя_n
тело__основной_функции;
}
тип имя_1(список_переменных)
{
тело_функции_1;
}
тип имя_2(список_переменных)
{
```

```
тело_функции_2;
}
...
тип имя_n(список_переменных)
{
тело_функции_n;
}
```

Здесь функции описаны после функции main(), однако до нее перечислены заголовки всех функций. Такого рода опережающие заголовки называют прототипами функций. Прототип указывает компилятору тип данных, возвращаемых функцией, тип переменных, выступающих в роли аргументов и порядок их следования. Прототипы используются для проверки правильности вызова функций в основной программе. Имена переменных, указанные в прототипе функции, компилятор игнорирует:

```
//Записи равносильны.
int func(int a,int b);
int func(int ,int);
```

Вызвать функцию можно в любом месте программы. Для вызова функции необходимо указать ее имя и в круглых скобках, через запятую перечислить имена или значения аргументов, если таковые имеются:

```
имя_функции(список_переменных);
```

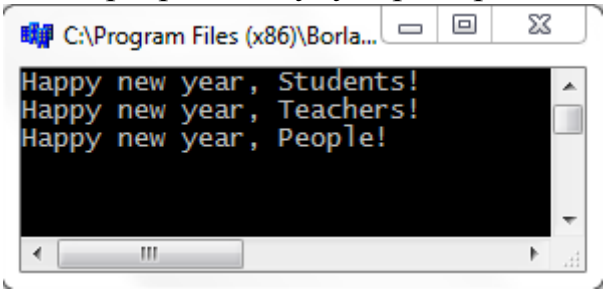
Рассмотрим пример. Создадим функцию f(), которая не имеет входных значений и не формирует результат. При вызове этой функции на экран выводится строка символов "Happy new year, ".

```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop

//---------------------------------

#pragma argsused
void f()  //Описание функции
        {
        cout << "Happy new year, ";
        }
int main(int argc, char* argv[])
{
        f();  //Вызов функции.
        cout <<"Students!" << endl;
        f(); //Вызов функции.
        cout <<"Teachers!" << endl;
        f(); //Вызов функции.
        cout <<"People!" << endl;
        getch();
}
```

Результатом работы программы будут три строки:



Далее приведен пример программы, которая пять раз выводит на экран фразу "`Hello World!`". Операция вывода строки символов оформлена в виде функции `fun()`. Эта функция так же не имеет входных значений и не формирует результат. Вызов функции осуществляется в цикле:

```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
//---------------------------------
#pragma argsused
 void fun()
{
cout << "Hello World!" << endl;
}

int main(int argc, char* argv[])
{
        for(int i=1;i<=5;fun(),i++);
        getch();
}
```

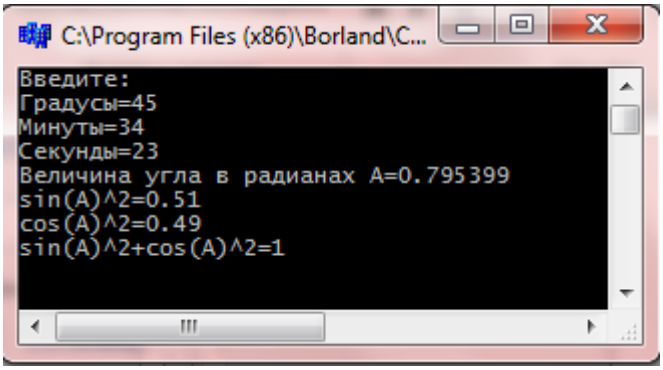
Если тип возвращаемого значения не void, то функция может входить в состав выражений. Типы и порядок следования переменных в определении и при вызове функции должны совпадать. Для того чтобы функция вернула какое-либо значение, в ней должен быть оператор:

`return (выражение);`

Далее приведен пример программы, которая проверяет значение выражения $\sin^2(\alpha)+\cos^2(\alpha)=1$ при заданном значении α. Здесь функция `radian` выполняет перевод градусной меры угла в радианную (Чтобы найти радианную меру какого-нибудь угла по заданной градусной мере, нужно помножить число градусов на $\pi/180$, число минут на $\pi/180\cdot60$, число секунд на $\pi/180\cdot60\cdot60$ и найденные произведения сложить).

```
//Ввод данных.
cout<<"Введите:"<<endl; //Величина угла:
cout<<"Градусы="; cin>>DEG;  //градусы,
cout<<"Минуты="; cin>>MIN;  //минуты,
cout<<"Секунды="; cin>>SEC;  //секунды.
//Величина угла в радианах.
RAD=radian(DEG,MIN,SEC); //Вызов функции.
cout << "Величина угла в радианах A="<<RAD << endl;
//Вычисление значения выражения и его вывод.
cout << "sin(A)^2="; cout << pow(sin(RAD),2) << endl;
cout << "cos(A)^2="; cout << pow(cos(RAD),2) << endl;
cout << "sin(A)^2+cos(A)^2=";
cout << pow(sin(RAD),2)+pow(cos(RAD),2) << endl;
getch();
}
SetConsoleCP(1251);
int DEG,MIN,SEC; double RAD;
```

Результат выполнения программы:



Переменные, описанные внутри функции, а так же переменные из списка аргументов, являются *локальными*. Например, если программа содержит пять разных функций, в каждой из которых описана переменная N, то для C++ это пять различных переменных. Область действия локальной переменной не выходит за рамки функции. Значения локальных переменных между вызовами одной и той же функции не сохраняются. Переменные, определенные до объявления всех функций и доступные всем функциям, называют *глобальными*. В функции глобальную переменную можно отличить, если не описана локальная переменная с теми же именем.

Глобальные переменные применяют для передачи данных между функциями, но это затрудняет отладку программы. Для обмена данными между функциями используют параметры функций и значения, возвращаемые функциями.

2.2 Передача параметров в функцию

Обмен информацией между вызываемой и вызывающей функциями осуществляется с помощью механизма передачи параметров. Список_переменных, указанный в заголовке функции называется *формальными*

параметрами или просто *параметрами функции*. Список_переменных в операторе вызова функции это *фактические параметры* или *аргументы*.

Механизм передачи параметров обеспечивает замену формальных параметров фактическими параметрами, и позволяет выполнять функцию с различными данными.

Между фактическими параметрами в операторе вызова функции и формальными параметрами в заголовке функции устанавливается взаимно однозначное соответствие. Количество, типы и порядок следования формальных и фактических параметров должны совпадать.

Передача параметров выполняется следующим образом. Вычисляются выражения, стоящие на месте фактических параметров. В памяти выделяется место под формальные параметры, в соответствии с их типами. Затем формальным параметрам присваиваются значения фактических. Выполняется проверка типов и при необходимости выполняется их преобразование. При несоответствии типов выдается диагностическое сообщение.

Передача параметров в функцию может осуществляться *по значению* и *по адресу*.

При передаче данных по значению функция работает с копиями фактических параметров, и доступа к исходным значениям аргументов у нее нет. При передаче данных по адресу в функцию передается не переменная, а ее адрес, и, следовательно, функция имеет доступ к ячейкам памяти, в которых хранятся значения аргументов. Таким образом, данные, переданные по значению, функция изменить не может, в отличие от данных, переданных по адресу.

Если требуется запретить изменение параметра внутри функции, используют модификатор const. Заголовок функции в общем виде будет выглядеть так:

```
тип имя_функции(const тип_переменной* имя_переменной, …)
```

Например:

```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#pragma hdrstop
//---------------------------------------------------------------
#pragma argsused
 int f1(int i)  //Данные передаются по значению
{
return(i++);
}
int f2(int* j) //Данные передаются по адресу.
//При подстановке фактического параметра,
//для получения его значения,
//применяется операция разадресации *.
{
return((*j)++);
}
int f3(const int* k)  //Изменение параметра не предусмотрено.
{
return(*k);
}

int main(int argc, char* argv[])
{
int a;
cout<<"a=";cin>>a;
f1(a);
cout<<"a="<<a<<"\n";
f2(&a);  //Для передачи фактического параметра
  //используется операция взятия адреса &.
cout<<"a="<<a<<"\n";
f3(&a);
cout<<"a="<<a<<"\n";
getch();
}
```

Результат работы программы:

Введено значение переменной a=5.

Значение переменной a после вызова функции f1 не изменилось: a=5.

Значение переменной a после вызова функции f2 изменилось: a=6

Значение переменной a после вызова функции f3 не изменилось: a=6

Удобно использовать передачу данных по адресу, если нужно чтобы функция изменяла значения переменных в вызывающей программе.

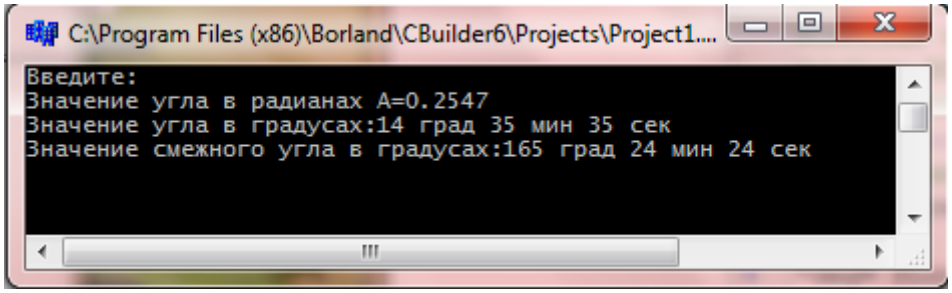
Далее приведен пример программы, в которой исходя из радианной меры некоторого угла вычисляется величина смежного с ним угла. На экран выводятся значения углов в градусной мере. Функция degree выполнят перевод из радианной меры в градусную (Чтобы найти градусную меру угла по заданной радианной нужно помножить число радиан на 180/π; если из полученной дроби выделить целую часть, то получим градусы; если из числа полученного умножением оставшейся дробной части 60 выделить целую часть получим минуты; секунды вычисляются аналогично из дробной части минут.). Эта функция ничего не возвращает. Ее аргументами являются значение

```
#include <vcl.h>
#include <iostream.h>
#include <conio.h>
#include <math.h>
#include <windows.h>
#define PI 3.14159
#pragma hdrstop
//---------------------------------------------------------
#pragma argsused
void degree (double rad, int *deg, int * min, int *sec)
{
*deg= floor(rad*180/PI);
*min=floor((rad*180/PI-(*deg))*60);
*sec=floor(((rad*180/PI-(*deg))*60-(*min))*60);
}

int main(int argc, char* argv[])
{ SetConsoleOutputCP(1251);
SetConsoleCP(1251);
 int DEG,MIN,SEC; double RAD;
cout<<"Введите:"<<endl;
cout << "Значение угла в радианах A=";cin>>RAD;
degree(RAD,&DEG,&MIN,&SEC);
cout << "Значение угла в градусах:";
cout<<DEG<<" град "<<MIN<<" мин "<<SEC <<" сек"<< endl;
degree(PI-RAD,&DEG,&MIN,&SEC);
cout << "Значение смежного угла в градусах:";
cout<<DEG<<" град "<<MIN<<" мин "<<SEC <<" сек"<< endl;
getch();
}
```

переменной rad, определяющее величину угла в радианах и адреса переменных deg, min, sec, в которых будут храниться вычисленные результаты — градусная мера угла.

Результат выполнения программы:



2.3. Возврат результата с помощью оператора return

Возврат результата из функции в вызывающую ее функцию осуществляется оператором `return (выражение);`

Работает оператор следующим образом. Вычисляется значение выражения, указанного после `return` и преобразуется к типу возвращаемого функцией значения. Выполнение функции завершается, а вычисленное значение передается в вызывающую функцию. Любые операторы, следующие в функции за оператором `return`, игнорируются. Программа продолжает свою работу с оператора, следующего за оператором вызова данной функции.

Оператор `return` может отсутствовать в функциях типа `void`, если возврат происходит перед закрывающейся фигурной скобкой, и в функции `main`.

Так же функция может содержать несколько операторов return, если это определено потребностями алгоритма. Например, в следующей программе, функция equation вычисляет корни квадратного уравнения. Если a=0 (уравнение не является квадратным), то в программу передается значение равное -1, если дискриминант отрицательный (уравнение не имеет действительных корней), то 1, а если положительный, то вычисляются корни уравнения и в программу передается 0.

```
    #include <iostream>
    #include <math.h>
    using namespace std;
    int equation( float a,float b,float c,float *x1,float
*x2)
    { float D=b*b-4*a*c;
    if (a==0) return -1;
    else if (D<0) return 1;
    else {
    *x1=(-b+sqrt(D))/2/a;
    *x2=(-b-sqrt(D))/2/a;
    return 0;
        }
    }
    int main()
    {
```

```
    float A, B, C, X1, X2; int P;
    cout<<"Enter the coefficients of the equation:"<<endl;
    cout<<"A="; cin>>A;
    cout<<"B="; cin>>B;
    cout<<"C="; cin>>C;
    P=equation( A, B, C, &X1, &X2);
    if (P==-1)  cout<<"input Error"<<endl;
    else if (P==1) cout<<"No real roots"<<endl;
    else  cout<<"X1="<<X1<<" X2="<<X2<<endl;
    return 0;
    }
```

3. РЕШЕНИЕ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ

Рассмотрим несколько задач с применением функций.

ЗАДАЧА 1. Вводится последовательность из N целых чисел, найти среднее арифметическое совершенных чисел и среднее геометрическое простых чисел последовательности.

Напомним, что целое число называется *простым*, если оно делится нацело только на само себя и единицу. Число называется *совершенным*, если сумма всех делителей, меньших его самого равна этому числу.

Для решения поставленной задачи понадобятся две функции:

- prostoe – определяет, является ли число простым, аргумент функции целое число N; функция возвращает 1, если число простое и 0 – в противном случае.;
- soversh – определяет, является ли число совершенным; входной параметр целое число N; функция возвращает 1, если число совершенным и 0 – в противном случае.

```
    #include <iostream>
    #include <math.h>
    unsigned int prostoe(unsigned int N) //Описание
функции.
    {
    //Функция определяет, является ли число простым.
    int i,pr;
    for(pr=1,i=2;i<=N/2;i++)
    if (N%i==0) {pr=0;break;}
    return pr;
    }
    unsigned int soversh(unsigned int N) //Описание
функции.
    {
    //Функция определяет, является ли число совершенным.
    unsigned int i,S;
    for(S=0,i=1;i<=N/2;i++)
    if (N%i==0) S+=i;  //Сумма делителей.
```

```
if (S==N) return 1;
else return 0;
}
using namespace std;
int main()
{
unsigned int i,N,X,S,kp,ks;
long int P;
cout <<"N="; cin>>N;
for(kp=ks=S=0,P=1,i=1;i<=N;i++)
{
cout <<"X="; cin >> X; //Вводится элемент
//последовательности.
if (prostoe(X)) // X – простое число.
{
kp++; //Счетчик простых чисел.
P*=X; //Произведение простых чисел.
}
if (soversh(X)) //X – совершенное число.
{
ks++; //Счетчик совершенных чисел.
S+=X; //Сумма совершенных чисел.
}
}
if (kp>0) //Если счетчик простых чисел больше нуля,
//считаем среднее геометрическое
//и выводим его,
cout<<"Geometric mean="<<pow(P,(float) 1/kp)<<endl;
else  //в противном случае – сообщение об
//отсутствии простых чисел.
cout<<"No prime numbers \n";
if (ks>0)  //Если счетчик совершенных чисел
//больше нуля, считаем среднее
//арифметическое и выводим его,
cout<<"Arithmetical mean="<<(float) S/ks<<endl;
else  //в противном случае – сообщение
//об отсутствии совершенных чисел.
cout<<"No perfect numbers \n";
return 0;
}
```

ЗАДАЧА 2. Вводится последовательность целых чисел, 0 – конец последовательности. Найти минимальное число среди простых чисел и максимальное, среди чисел, не являющихся простыми.

Для решения данной задачи создадим функцию prostoe, которая будет проверять, является ли число N простым. Входным параметром функции будет целое положительное число N. Функция будет возвращать значение 1, если число простое и 0 – в противном случае.

Текст программы:

```
#include <iostream>
using namespace std;
unsigned int prostoe(unsigned int N)
{
int i,pr;
for(pr=1,i=2;i<=N/2;i++)
if (N%i==0) {pr=0;break;}
return pr;
}
int main()
{
int kp=0,knp=0,min,max,N;
for (cout << "N=", cin>>N; N!=0; cout<<"N=", cin>>N)
//В цикле вводится элемент последовательности N.
if (prostoe(N)) //N – простое число,
{
kp++; //счетчик простых чисел.
if (kp==1) min=N; //Предполагаем, что первое
//простое число минимально,
else if (N<min) min=N; //если найдется меньшее
//число, сохраняем его.
}
else //N – простым не является,
{
knp++; //счетчик чисел
//не являющихся простыми.
if (knp==1) max=N; //Предполагаем, что первое
//не простое число
//максимально,
else if (N>max) max=N; //если найдется большее
//число, сохраняем его.
}
if (kp>0)  //Если счетчик простых
//чисел больше нуля,
cout <<"min= "<<min<<"\t"; //выводим значение
//минимального простого
//числа,
else  //в противном случае –
cout <<"Net prostih"; //сообщение об
//отсутствии простых
```

```
//чисел.
if (knp>0)  //Если счетчик чисел
//не являющихся простыми
//больше нуля,
cout <<"max="<<max<<endl; //выводим значение
//максимального числа,
else  //в противном случае –
cout <<"Net ne prostih"; //сообщение об
//отсутствии чисел
//не являющихся
//простыми.
return 0;
}
```

ЗАДАЧА 3. Вводится последовательность из N целых положительных чисел. В каждом числе найти наименьшую и наибольшую цифры.

Программный код:

```
#include <iostream>
using namespace std;
unsigned int Cmax(unsigned long long int P)
{
unsigned long long int M,Z;
unsigned int max;
for (int i=1, M=P; M/10>0;M/=10, Z=M)
{
if (i==1) {max=M%10;i++;}
if (M%10>max) max=M%10;
}
if (Z>max) max=Z; //Проверка последней цифры числа.
return max;
}
unsigned int Cmin(unsigned long long int P)
{
unsigned long long int M,Z;
unsigned int min;
for (int i=1, M=P; M/10>0;M/=10, Z=M)
{
if (i==1) {min=M%10;i++;}
if (M%10<min) min=M%10;
}
if (Z<min) min=Z;  //Проверка последней цифры числа.
return min;
}
int main()
{
unsigned int N, k;
```

```
unsigned long long int X;
for (cout<<"N=",cin>>N,k=1;k<=N;k++)
{
cout<<"X=";cin>>X;
cout<<"CifraMax="<<Cmax(X);
cout<<"  CifraMin="<<Cmin(X)<<endl;
}
return 0;
}
```

ЗАДАЧА 4. Вводится последовательность целых положительных чисел, 0 — конец последовательности. Определить сколько в последовательности чисел — палиндромов (Палиндром — любой симметричный относительно своей середины набор символов.).

Программный код:

```
#include <iostream>
using namespace std;
int kol(unsigned long long int N)
{ //Функция возвращает количество цифр в числе N.
int k;
for (k=1;N/10>0; k++,N/=10);
return k;
}
int high(unsigned long long int N)
{ //Функция возвращает старший разряд числа N.
//Если N=12345, то r=10000.
unsigned long long int r;
for (r=1;N/10>0; r*=10,N/=10);
return r;
}
bool palindrom(unsigned long long int N)
{   //Функция   определяет   является   ли   число   N
палиндромом,
//возвращает true, если N - палиндром
//и false, в противном случае
unsigned long long int M; int i;
bool Fl=true;
for (M=N,i=1;i<kol(N)/2;M%=high(M),M/=10,i++)
if (M%10 != M/high(M)) { Fl=false; break; }
return Fl;
}
int main()
{ unsigned long long int X;
int K;
for (K=0,cout<<"X=",cin>>X;X!=0;cout<<"X=",cin>>X)
if (palindrom(X)) K++;
```

```
cout<<"The number of palindromic numbers K="<<K<<endl;
return 0;
}
```

ЗАДАЧА 5. Заданы два числа - X в двоичной системе счисления, Y в системе счисления с основанием пять. Найти сумму этих чисел. Результат вывести в десятичной системе счисления.

Любое целое число N, заданное в b-ичной системе счисления, можно записать в виде:

$N=P_n \cdot b^n + P_{n-1} \cdot b^{n-1} + ... + P_2 \cdot b^2 + P_1 \cdot b + P_0 = \sum_{i=0}^{n} P_i b^i$

где b – основание системы счисления (целое положительное фиксированное число),

P_i – разряд числа:

$0 \le P_i \le b-1, i=0,1,2,...,n.$

Например,

$743_{10} = 7 \cdot 10^2 + 4 \cdot 10^1 + 3 \cdot 10^0 = 700 + 40 + 3 = 743_{10}$

$1011101_2 = 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 64 + 16 + 8 + 4 + 1 = 93_{10}$

Создадим функцию для перевода целого числа N заданного в b-ичной системе счисления в десятичную систему счисления.

```
#include <iostream>
using namespace std;
unsigned long long int DecNC(unsigned long long int N,
unsigned int b)
{ //Функция выполняет перевод числа N,
//заданного в b-ичной системе счисления,
//в десятичную систему счисления
unsigned long long int S,P; int i;
for (S=0,P=1,i=1;N/10>0;S+=N%10*P,P*=b,N/=10);
return S+=N*P;
}
int main()
{
unsigned long long int X,Y; unsigned int bX,bY;
cout<<"X=";cin>>X;  //Ввод числа X.
cout<<"b=";cin>>bX; //Ввод основания с/с.
cout<<"Y=";cin>>Y;  //Ввод числа X.
cout<<"b=";cin>>bY; //Ввод основания с/с.
//Вывод заданных чисел в десятичной с/с.
cout<<X<<"("<<bX<<")="<<DecNC(X,bX)<<"(10)"<<endl;
cout<<Y<<"("<<bY<<")="<<DecNC(Y,bY)<<"(10)"<<endl;
//Вычисление суммы и вывод результата.
cout<<X<<"("<<bX<<")+"<<Y<<"("<<bY<<")=";
cout<<DecNC(X,bX)+DecNC(Y,bY)<<"(10)"<<endl;
return 0;
}
```

ЗАДАЧА 6. Задано число X в десятичной системе счисления. Выполнить перевод числа в системы счисления с основанием 2, 5 и 7.

Вообще, для того чтобы перевести целое число из одной системы счисления в другую необходимо выполнить следующие действия:

1. поделить данное число на основание новой системы счисления;
2. перевести остаток от деления в новую систему счисления; получается младший разряд нового числа;
3. если частное от деления больше основания новой системы, продолжать деление, как указано в п.1; новый остаток, переведенный в новую систему счисления, дает второй разряд числа и т. д.

На рис. 12 приведен пример перевода числа 256, заданного в десятичной

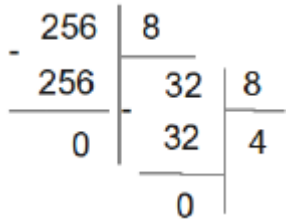


Рис. 14. Пример перевода числа в новую систему счисления

системе счисления, в восьмеричную. В результате получим, $256_{10}=400_8$.

Далее приведен текст программы, реализующей решение задачи 4.6.

```
#include <iostream>
unsigned long long int NC(unsigned long long int N,
 unsigned int b)
{ unsigned long long int S,P;
for (S=0,P=1;N/b>b;S+=N%b*P,P*=10,N/=b);
return S+=N%b*P+N/b*P*10;
}
int main()
{ unsigned long long int X;
cout<<"X=";cin>>X;  //Ввод числа X.
//Перевод числа X в заданные системы счисления.
cout<<X<<"(10)="<<NC(X,2)<<"(2)"<<endl;
cout<<X<<"(10)="<<NC(X,5)<<"(5)"<<endl;
cout<<X<<"(10)="<<NC(X,7)<<"(7)"<<endl;
return 0;
}
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1 Применение функций при работе с последовательностями чисел

Разработать программу на языке С++ для следующих заданий:

1. Вводится последовательность целых положительных чисел, 0 – конец последовательности. Для каждого элемента последовательности определить и вывести на экран число, которое получится после записи цифр исходного числа в обратном порядке.

2. Вводится последовательность целых чисел, 0 – конец последовательности. Определить содержит ли последовательность хотя бы одно совершенное число. Совершенное число равно сумме всех своих делителей, не превосходящих это число. Например, 6=1+2+3 или 28=1+2+4+7+14.

3. Вводится последовательность из N целых положительных элементов. Определить содержит ли последовательность хотя бы одно простое число. Простое число не имеет делителей, кроме единицы и самого себя.

4. Вводится последовательность из N целых положительных элементов. Посчитать количество чисел палиндромов. Числа палиндромы симметричны относительно своей середины, например, 12021 или 454.

5. Вводится последовательность из N целых положительных элементов. Подсчитать количество совершенных и простых чисел в последовательности.

6. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Определить в каком из чисел больше всего делителей.

7. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Определить в каком из чисел больше всего цифр.

8. Вводится последовательность из N целых положительных элементов. Проверить содержит ли последовательность хотя бы одну пару соседних дружественных

чисел. Два различных натуральных числа являются дружественными, если сумма всех делителей первого числа (кроме самого числа) равна второму числу. Например, 220 и 284, 1184 и 1210, 2620 и 2924, 5020 и 5564..

9. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Для элементов последовательности, находящихся в диапазоне от единицы до m вычислить и вывести на экран соответствующие числа Фибоначчи. Здесь m — целое положительное число, которое необходимо ввести.

10. Вводится последовательность из N целых положительных элементов. Найти число с минимальным количеством цифр.

11. Вводится последовательность из N целых элементов. Для всех положительных элементов последовательности вычислить значение факториала. Вывести на экран число и его факториал.

12. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Вывести на экран все числа последовательности являющиеся составными и их делители. Составное число имеет более двух делителей, то есть не является простым.

13. Вводится последовательность из N целых положительных элементов. Определить содержит ли последовательность хотя бы одно число Армстронга. Число Армстронга — натуральное число, которое равно сумме своих цифр, возведенных в степень, равную количеству его цифр. Например, десятичное число 153 —число Армстронга, потому что: $1^3+3^3+5^3=1+27+125=153$.

14. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Найти среднее арифметическое простых чисел в этой последовательности. Простое число не имеет делителей, кроме единицы и самого себя.

15. Вводится последовательность из N целых положительных элементов. Определить сколько в последовательности пар соседних взаимно простых чисел. Различные натуральные числа являются взаимно простыми, если их наибольший общий делитель равен единице.

16. В последовательности из N целых положительных элементов найти сумму всех недостаточных чисел. Недостаточное число всегда больше суммы всех своих делителей за исключением самого числа.

17. Вводится последовательность из N целых положительных элементов. Посчитать количество элементов последовательности, имеющих в своем представлении цифру 0.

18. Вводится N пар целых положительных чисел a и b. В случае, если a > b вычис лить:

$$C = \frac{a!}{b!\cdot(a-b)!}$$

19. Вводится последовательность из N целых элементов. Для каждого элемента последовательности найти среднее значение его цифр.

20. Вводится последовательность целых положительных чисел, 0 – конец последовательности. Для каждого элемента последовательности определить и вывести на экран число, которое получится, если поменять местами первую и последнюю цифры исходного числа.

21. Вводится последовательность из N целых элементов. Для каждого элемента последовательности вывести на экран количество цифр и количество делителей.

22. Вводится последовательность из N целых положительных элементов. Среди элементов последовательности найти наибольшее число - палиндром. Числа палиндромы симметричны относительно своей середины, например, 12021 или 454.

23. Поступает последовательность целых положительных чисел, 0 – конец последовательности. Для каждого элемента последовательности вывести на экран сумму квадратов его цифр.

24. Вводится последовательность из N целых положительных элементов. Для простых элементов последовательности определить сумму цифр. Простое число не имеет делителей, кроме единицы и самого себя.

25. Вводится последовательность целых положительных чисел, 0 – конец последовательности. Среди элементов последовательности найти наименьшее составное число. Составное число имеет более двух делителей, то есть не является простым.

2 Применение функций для вычислений в различных системах счисления

Разработать программу на языке C++ для решения следующей задачи.

Заданы два числа — A и B, первое в системе счисления с основанием p, второе в системе счисления с основанием q. Вычислить значение C по указанной формуле и вывести его на экран в десятичной системе счисления и системе счисления с основанием r. Исходные данные для решения задачи представлены в табл. 4.1.

Вариант	**p**	**q**	**C**	**r**
1.	2	8	$A^2 \cdot (A + B)$	3
2.	3	7	$2 \cdot (A^2 + B^2)$	4
3.	4	6	$2 \cdot B^2 \cdot (A + B)$	5
4.	5	5	$(A - B)^2 + 3 \cdot A$	6
5.	6	4	$A^2 + A \cdot B$	7
6.	7	3	$(5 \cdot B - 2 \cdot A)^2$	8
7.	8	2	$(2 \cdot A - 3 \cdot B)^2$	5
8.	3	8	$(B - A)^2 + 2 \cdot A$	6
9.	4	7	$B^3 - B^2 + 2 \cdot A$	2
10.	5	6	$A^3 - A^2 + 3 \cdot B$	8
11.	6	5	$(4 \cdot A - 5 \cdot B)^2$	3
12.	7	4	$A^2 + 2 \cdot A + B^2$	5
13.	8	3	$A^2+3 \cdot B+B^2$	7
14.	4	2	$A^2-2 \cdot A+B$	6
15.	5	8	$3 \cdot B^2-2 \cdot B+A$	3
16.	6	7	$A^2+(B-A)^2$	2
17.	7	6	$3 \cdot B^2+2 \cdot A \cdot B$	8
18.	8	5	$2 \cdot A^2+3 \cdot A \cdot B$	7
19.	2	4	$B^3-2 \cdot B+A$	3
20.	3	8	$A^3-2 \cdot A+B$	4
21.	4	7	$(5 \cdot A-2 \cdot B)^2$	5
22.	5	6	$(B^2-3 \cdot A)^2$	7
23.	6	5	$(A^2-2 \cdot B)^2$	8
24.	7	4	$A^2 \cdot B^2-A \cdot B$	6
25.	8	3	$A \cdot B+A^2-B$	2

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Архангельский А.Я. Программирование в C++ Builder 6. – М.: Издательство БИНОМ, 2003. – 1152 с.
2. Архангельский А.Я. C++ Builder 6. Справочное пособие. Книга 2. Классы и компоненты.– М.: Бином-Пресс, 2002. – 528 с.
3. Вальпа О.Д. Borland C++ Builder. Эксперсс-курс. – СПб.: БХВ-Петербург, 2006. – 224 с.
4. Вирт Н. Алгоритмы и структуры данных. – М.: Мир, 1989. – 360 с.
5. Глушаков С.В., Коваль А.В., Смирнов С.В. Язык программирования C++. Учебный курс. Харьков: “Фолио”, Ростов-на-Дону: “Феникс”, 2001.
6. Культин Н.Б. C++ Builder в задачах и примерах. – СПб.: БХВ-Петербург, 2005. – 336 с.
7. Павловская Т.А., Щупак А.Ю. C/C++. Структурное программирование: Практикум. – СПб.: Питер, 2003. – 240 с.
8. Прата Стивен. Язык программирования C++. Лекции и упражнения. Учебник: - СПб.: ООО “ДиаСофтЮП”, 2005. -1104 с.
9. Федоренко Ю.П. Алгоритмы и программы на C++ Builder. – М.: ДМК Пресс, 2010. – 544 с.