ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИЙ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ
РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ
ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Курсовая работа

По предмету Программирование на C++

На тему: Ввод и вывод на языке C++

Студента 2в курса направления «Телекоммуникационные технологии»

Нурниязов К.

Преподаватель: Ядгаров Ш.

Нукус 2014

План

Введение

1.1 Теоретические сведения

1.2 Структура программы

1.2 Ввод данных

1.3 Операторы и выражения

Практический часть

Заключение

Литературы

Введение

«Потоковый ввод/вывод» — содержит подробное изложение
стандартных возможностей ввода/вывода C++. В главе обсуждается диапазон
методик вода/вывода, достаточный для решения большинства задач, связанных с вводом/выводом, и дается обзор остальных возможностей. Многие из обсуждаемых средств ввода/вывода являются объектно-ориентированными. Такой стиль ввода/вывода использует многие другие элементы языка, такие, как ссылки, перегрузка функций и перегрузка операций. Рассматриваются самые различные аспекты ввода/вывода в C++, включая вывод с помощью операции передачи в поток, ввод с помощью операции извлечения из потока, безопасный по типу ввод/вывод, форматированный ввод/вывод, неформатируемый ввод/вывод (позволяющий повысить производительность). Пользователи могут специфицировать, каким образом должен осуществляться ввод/вывод определяемых пользователем типов, перегружая операцию передачи в поток («) и извлечения из потока (»). Расширяемость является одной из наиболее ценныхособенностей C++. Для выполнения задач форматирования в C++ предусмотрены различные манипуляторы. В главе обсуждаются потоковые манипуляторы,
позволяющие выводить целые числа в различных системах счисления, управлять точностью чисел с плавающей точкой, устанавливать ширину полей вывода, отображать десятичную точку и конечные нули, выравнивать вывод, устанавливать и сбрасывать состояния формата, задавать заполняющие символы полей. Мы также представляем пример, в котором создается определяемый пользователем манипулятор выходного потока.

Теоретические сведения

Структура программы

В языке C++ любая программа состоит из одной или более функций, задающих действия, которые нужно выполнить. Выполнение любой программы начинается с функции main. Далее идет текст программы, заключенный в фигурные скобки. Таким образом, структура программы имеет вид:

```
main ( )
{
Тело программы
}
```

В самом простом случае функция main не имеет аргументов, поэтому в скобках ничего не содержится. Для работы программы, обеспечивающей ввод и вывод информации, перед функцией main необходимо поместить строку:

```
#include <stdio.h>
```

Алфавит языка и типы данных

Алфавит языка включает латинские прописные и строчные буквы, цифры и специальные знаки. К последним относятся: . (точка), , (запятая), ` (апостроф), : (двоеточие) и др.

Важным понятием языка является идентификатор, который используется в качестве имени объекта, например, переменной, функции и т.п. Идентификатор может содержать до 32 символов и состоит из букв и цифр, но начинается обязательно с буквы. Строчные буквы отличаются от прописных, поэтому идентификаторы SIGMA и sigma считаются разными.

В языке СИ существует несколько типов данных. Каждый тип данных определяется одним из следующих ключевых слов:

int (целый) - задает значения, к которым относятся все целые числа. Диапазон возможных целых значений лежит в пределах от -32768 до 32767, переменная типа int занимает 16 бит;

short (короткий целый) - соответствующие объекты не могут быть больше, чем int, переменные этого типа занимают 16 бит;

long (длинный целый) - соответствующие объекты не могут быть меньше, чем int. Переменная типа long занимает 32 бита и позволяет представить целые числа от -2147483648 до 2147483647;

char (символьный) - задает значения, которые представляют различные символы;

unsigned (беззнаковый) - в языке СИ можно объявлять некоторые типы (char, short, int, long) беззнаковыми с помощью модификатора unsigned (например, unsigned short). Это значит, что соответствующие переменные не будут иметь отрицательных значений. В результате они могут принимать большие положительные значения, чем переменные знаковых типов. В случае типа int объявления вида «unsigned int a;» можно записать «unsigned a;»;

float (вещественный) - задает значения, к которым относятся вещественные числа, имеющие дробную часть, отделяемую точкой. Вещественные числа могут быть записаны также в экспоненциальной форме. Например, -1.58e+2 (что равно $-1{,}58 \cdot 10^2$). В языке СИ переменная типа float занимает 32 бита. Она может принимать значения в диапазоне от +3.4e-38 до +3.4e+38;

double (двойная точность) - определяет вещественные переменные, занимающие в два раза больше места, чем переменная типа float. Переменная типа double занимает 64 бита. Она может принимать значения в диапазоне от +1.7e-308 до +1.7e+308.

Ввод и вывод информации

Форматный вывод

Вначале рассмотрим функцию, определяющую форматный вывод:

printf("управляющая строка", аргумент1, аргумент2, ...);

Управляющая строка содержит объекты трех типов: обычные символы, которые просто выводятся на экран дисплея, спецификации преобразования, каждая из которых вызывает вывод на экран значения очередного аргумента из последующего списка и управляющие символы-константы.

Каждая спецификация преобразования начинается со знака % и заканчивается некоторым символом, задающим преобразования.

Символ преобразования связан с типом переменных. приведем символы преобразования:

d - значением аргумента является десятичное целое число;

o - значением аргумента является восьмеричное целое число;

x - значением аргумента является шестнадцатеричное целое число;

c - значением аргумента является символ;

s - значением аргумента является строка символов;

e - значением аргумента является вещественное число в экспоненциальной форме;

f - значением аргумента является вещественное десятичное число с плавающей точкой;

u - значением аргумента является беззнаковое целое число;

p - значением аргумента является указатель (адрес).

Если после знака % записан не символ, то он выводится на экран. Функция printf использует управляющую строку, чтобы определить, сколько всего аргументов и каковы их типы.

Например, в результате работы программы получены переменная i, имеющая значение 100, и переменная j, имеющая значение 25. Обе переменные целого типа. Для вывода этих переменных на экран в виде

i=100 j=25

необходимо применить функцию

printf(“i=%d j=%d”,i,j);

Как было описано выше, в кавычках задается формат вывода. перед знаком % записываются символы, которые будут непосредственно выданы на экран. После знака % применена спецификация d, т.к. переменные i и j имеют целый тип. Сами i и j приведены через запятую в списке аргументов. Если результат должен быть представлен в виде

i=100; j=25

необходимо применить функцию

printf(“i=%d; j=%d, i, j);

Если после знака % стоит цифра, то она задает поле, в котором будет выполнен вывод числа. Приведем несколько функций printf, которые будут обеспечивать вывод одной и той же переменной S целого типа, имеющей значение 336.

Функция printf(“%2d”, S); выдает на экран:

336

В этом примере ширина поля (она равна двум) меньше, чем число цифр в числе 336, поэтому поле автоматически расширяется до необходимого размера.

Функция printf(“%6d”, S);

выдаст на экран:

_ _ _336

(6 позиций)

То есть, в результате работы функции число сдвинуто к правому краю поля, а лишние позиции перед числом заполнены пробелами.

Функция printf(“%-6d”, S);

выдаст на экран:

336_ _ _

(6 позиций)

Знак «минус» перед спецификацией приводит к сдвигу числа к левому краю поля.

Рассмотрим вывод вещественных чисел.

Если перед спецификацией f ничего не указано, то выводится число с шестью знаками после запятой. при печати числа с плавающей точкой перед спецификацией f тоже могут находиться цифры.

Рассмотрим на конкретном примере три возможные ситуации:

%6f - печать числа с плавающей точкой в поле из шести позиций;

%.2f - печать числа с плавающей точкой с двумя цифрами после десятичной точки;

%6.2f - печать числа с плавающей точкой в поле из шести позиций и двумя цифрами после десятичной точки.

Например, в результате работы программы получены переменные вещественного типа a=3,687 и b=10,17.

Если для вывода значений использована функция

printf(“%7f %8f”,a,b);

то результат будет представлен в виде строки:

_ _ 3.687 _ _ _ _10.17

(7 поз.) (8 позиций)

Как видно из примера, лишние позиции заполняются пробелами. Если для вывода значений использована функция

printf(“%.2f %/2f”, a, b);

то результатом будет строка:

3.69 10.17,

из которой следует, что в первом числе третья цифра после десятичной точки отброшена с округлением, т.к. указан формат числа с двумя цифрами после десятичной точки.

Если для вывода значений использована функция

printf(“%7.2f e”,a,b);

то будет выведена строка:

_ _ _ 3.681.010000e+01

(7 позиций)

Поскольку для вывода значения переменной b применена спецификация e, то результат выдан в экспоненциальной форме. Следует отметить, что , если ширина поля меньше, чем число цифр в числе, то поле автоматически расширяется до необходимого размера.

Как было отмечено выше, в управляющей строке могут содержаться управляющие символьные константы. Среди управляющих символьных констант наиболее часто используются следующие:

\a - для кратковременной подачи звукового сигнала;

\b - для перевода курсора влево на одну позицию;

\n - для перехода на новую строку;

\r - для перевода курсора в начало текущей строки;

\t - для горизонтальной табуляции;

\v - для вертикальной табуляции.

Предположим, в результате работы программы переменная i получила значение 50. В результате записи инструкции вызова функции

printf(“\t ЭВМ\n%d\n”,i);

сначала выполнится горизонтальная табуляция (\t), т.е. курсор сместится от края экрана на 8 позиций, затем на экран будет выведено слово “ЭВМ”, после этого курсор переместится в начало следующей строки (\n), затем будет выведено целое значение i по формату d, и окончательно курсор перейдет в начало новой строки (\n). Таким образом, результат работы этой функции на экране будет иметь вид:

_ _ _ _ _ _ _ _ ЭВМ

Ввод данных

Для форматного ввода данных используется функция

scanf(«управляющая строка», аргумент1, аргумент2,...);

Если в качестве аргумента используется переменная, то перед ее именем записывается символ &.

Управляющая строка содержит спецификации преобразования и используется для установления количества и типов аргументов. спецификации для определения типов аргументов такие же, как и для функции printf. Перед символами d,o,x,f может стоять буква l. В первых трех случаях соответствующие переменные должны иметь тип long, а в последнем double.

Рассмотрим пример. Требуется ввести значения для переменных i (целого типа) и a (вещественного типа). Эту задачу выполнит функция:

scanf(“%d%f”,&i,&a);

В управляющей строке спецификации трех типов могут быть отделены друг от друга различными знаками, в том числе и пробелом. Следовательно, при занесении значений переменных необходимо использовать указанный разделитель. Если спецификации не отделены одна от другой никакими значениями, то значения переменных заносятся через пробел.

В языке СИ есть две очень удобные функции puts и gets, позволяющие вводить и выводить строку символов. Пример их использования показан ниже:

```
#include<stdio.h>

main()

{
```

```
char q[40]; /*объявление строки символов*/

puts(“Введите строку символов”);

gets(q); /*ввод строки символов*/

puts(q); /*вывод строки символов*/

}
```

В результате работы программы вначале на экране появится текст:

Введите строку символов,после чего следует ввести какую-либо строку символов. Эта информация при помощи оператора gets будет присвоена элементам символьного массива q. Оператор puts выведет строку символов.

Операторы и выражения

Выражения широко используются в программах на языке СИ и представляют собой формулы для вычисления переменных. Они состоят из операндов (переменные, константы и др.), соединенных знаками операций (сложение, вычитание, умножение и др.). Порядок выполнения при вычислении значения выражения определяется их приоритетами и может регулироваться с помощью круглых скобок. Наиболее часто арифметические выражения используются в операторе присваивания. Этот оператор заменяет значение переменной в левой части оператора на значение выражения, стоящего в правой части, и имеет следующую форму:

переменная = выражение;

В языке СИ может быть использован модификатор const, запрещающий какие бы то ни было переопределения константы: ее уменьшение, увеличение и т.п. Модификатор const, используемый отдельно, эквивалентен const int. Приведем примеры:

const float a=3.5;

const j=47;

Результатом деления по модулю является остаток от деления. Например, если b=5, c=2, то при выполнении операции

a=b%c,

переменная a получит значение 1.

Широкое распространение находят также выражения с еще одной нетрадиционной терпарной операцией ?: . В выражении

y=x?a:b,

y=a, если x не равно нулю, и y=b, если x равно нулю. Следующее выражение

y=(a>b)?a:b;

позволяет присвоить переменной y значение большей переменной (a или b), т.е. y=max(a,b).

В таблице 2 приведены некоторые функции, применяемые при программировании на СИ.

Таблица 2

ческая	языке СИ
	X)
	loat X)
	s(double X)

ı(double X)

ı(double X)

(double X)

double X)

(double X)

(double X)

(double X)

l0(double X)

(double X)

/(double X, double Y)

Перед аргументом и функцией указан допустимый тип (при программировании эта запись типа опускается).

В программах на языке СИ важная роль отводится комментариям, которые повышают наглядность и удобство чтения программ. Они могут быть записаны в любом месте программы и обрамляются символами /* и */.

Рассмотрим пример программы на языке СИ.

Требуется вычислить:

Для работы с математическими функциями необходимо перед функцией main поместить строку:

#include <math.h>

Программа на СИ имеет вид:

#include <stdio.h>

```
#include <math.h>

main()

{

float z,f,k; /*объявление вещественных переменных z,f,k*/

double y,a,b,c,d,x; /*объявление переменных y,a,b,c,d,x переменными двойной точности*/

scanf("%f %f %f %lf %lf", &z, &f, &k, &d, &x); /* ввод с клавиатуры переменных z,f,k,d,x*/

a=log(x)+(z+f)/k;

b=sin(x)+tan(x);

c=pow(d+exp(x),1./5);

y=(a+b)/c;

printf("%lf %lf %ef %lf", a, b, c, y); /*вывод на экран значений переменных a,b,c,y*/

}
```

Следует обратить внимание на то, что при вычислении переменной c, выражение, стоящее в правой части, представлено как $\sqrt[5]{(d+e^x)}$, поэтому применена функция pow. Еще одно замечание. Следует осторожно подходить к делению целых чисел. Если оба операнда целые, то результат тоже будет целым, а дробная часть отбрасывается. таким образом, при выполнении операции 1/5, результат будет равен нулю. Для того чтобы сохранить дробную часть, хотя бы один из операндов должен быть вещественным. Это условие выполнено при вычислении 1./5 .

Из таблицы 3 взять задание по варианту и написать программу для вычисления выражения на языке СИ.

"Операторы цикла в языке программирования Си++"

Цель работы: ознакомиться с циклическими алгоритмами и операторами, реализующими эти алгоритмы. Освоить особенности применения каждого оператора. Составить программы с использованием всех операторов цикла.

1. Теоретические сведения

Оператор цикла while

Описание: while (выражение) оператор;

Действие:

Выполняется оператор до тех пор, пока значение выражения в скобках истинно. Проверка значения выражения происходит перед каждым выполнением оператора. Когда значение выражения ложно, цикл while заканчивается. Если выражение ложно с самого начала, оператор не выполняется ни разу.

Комментарий:

Следует заметить, что после ключевого слова while и выражения, заключенного в круглые скобки, точка с запятой не ставится.

Оператор иногда называется телом цикла. В теле цикла должны выполняться действия, в результате которых меняется значение управляющего выражения. В противном случае можем получить бесконечный цикл.

Пример:

/*Демонстрация цикла while*/

```
#include <stdio.h>

main( )

{

int i=1

while (getchar()!='R') i++;

/*оператор getchar() вводит любой символ с клавиатуры*/

printf("Символ R %d-й",i);

}
```

Приведенная программа позволяет определить порядковый номер первой введенной буквы R в последовательности символов. Она показывает использование цикла while, в теле которого всего одна инструкция (i++ - увеличение значения целого числа i на единицу). Если запустить эту программу на выполнение и ввести последовательность символов, например: abFk!Rgm, то на экране появится строка: Символ R 6-й.

Оператор цикла do-while

Описание: do оператор while (выражение);

Действие: В операторе do-while тело цикла выполняется по крайней мере один раз. Тело цикла будет выполняться до тех пор, пока выражение в скобках не примет ложное значение. Если оно ложно при входе в цикл, то его тело выполняется ровно один раз.

Комментарий: после слова while и выражения, заключенного в скобки, ставится точка с запятой. Если в теле цикла содержится более одной инструкции, то операторы цикла заключаются в фигурные скобки.

Пример:

```
/*Демонстрация цикла do-while */

#include <stdio.h>

main()

{

int i=0; /*i=0, а не единице*/

do i++; while (getchar()!='R');

printf("Символ R %d-й",i);

}
```

Программа, представленная выше, теперь написана с циклом do-while. Результат программы будет таким же.

Оператор цикла for

Описание: for (выражение 1; выражение 2; выражение 3) оператор;

Действие:

В круглых скобках содержится три выражения. Первое из них служит для инициализации счетчика. Она осуществляется только один раз - когда цикл for начинает выполняться. Второе выражение необходимо для проверки условия, которая осуществляется перед каждым возможным выполнением тела цикла. Когда выражение становится ложным, цикл завершается. Третье выражение вычисляется в конце каждого выполнения тела цикла, происходит приращение числа на шаг.

Комментарий: в операторе цикла for точка с запятой после закрывающейся круглой скобки не ставится. Любое из трех или все три выражения в операторе могут отсутствовать, однако разделяющие их точки с запятыми опускать нельзя. Если отсутствует выражение 2, имеем бесконечный цикл. Например:for (scanf("%d",&p);;p++) оператор;

В языке СИ предусмотрены две нетрадиционные операции: (++) - для увеличения на единицу и (--) - для уменьшения на единицу значения операнда. Операции ++ и -- можно записывать как перед операндом, так и после него. В первом случае (++n или --n) значение операнда (n) изменяется перед его использованием в соответствующем выражении, а во втором (n++ или n--) - после его использования.

Если отсутствуют выражения 1 и 3, цикл становится эквивалентным while. Например:for (;a<20;) оператор;

Каждое из выражений может состоять из нескольких выражений, объединенных операцией "запятая". Например: for(i=0, j=1; i<100; i++, j++) a[i]=b[j];

Тело цикла заключается в фигурные скобки, если в нем более одного оператора.

Пример:

```
/*демонстрация цикла for*/

#include <stdio.h>

main()

{int i,j=1,k;

for (i=1;i<=3;i++)

printf("Минск\t");

/*В цикле for три раза выполняется функция вывода*/
```

```
/*Здесь i-управляющая переменная цикла*/

printf("\nУкажите число повторений цикла\n");

scanf("%d",&k);

for (i=1;i<=k;i++)

{j*=i;

printf("%d",j);}

/*Здесь две инструкции (более одной), поэтому они заключаются в фигурные скобки*/

j=i;

printf("\n");

/*Переменной j присваивается значение 1 и осуществляется перевод курсора*/

/*В следующем цикле for выполняются те же действия, что и в предыдущем*/

for (i=1;i<=k;i++) printf("%d ", j*=i);

}
```

Результаты выполнения программы следующие:

Минск Минск Минск

Укажите число повторений цикла; 5

1 2 6 24 120

1 2 6 24 120

Оператор break

Описание:

Break используется для прекращения выполнения цикла из-за обнаружения ошибки, для организации дополнения к условию в заголовке цикла, для прекращения бесконечного цикла.

Пример:

```
while (st>0 && st<25)

{

if st==4||st==8||st==12)

break;

}
```

Работа цикла полностью прекращается, как только условие в операторе if становится истинным.

Оператор continue

Действие: Этот оператор может использоваться во всех трех типах циклов. Как и в случае оператора break, он приводит к изменению характера выполнения программы. Однако вместо завершения работы цикла наличие оператора continue вызывает пропуск "оставшейся" части итерации и переход к началу следующей.

Пример. Заменим в предыдущей программе оператор break на continue.

```
while (st>0 && st<25)

{
```

```
if (st==4||st==8||st==12)

continue;

}
```

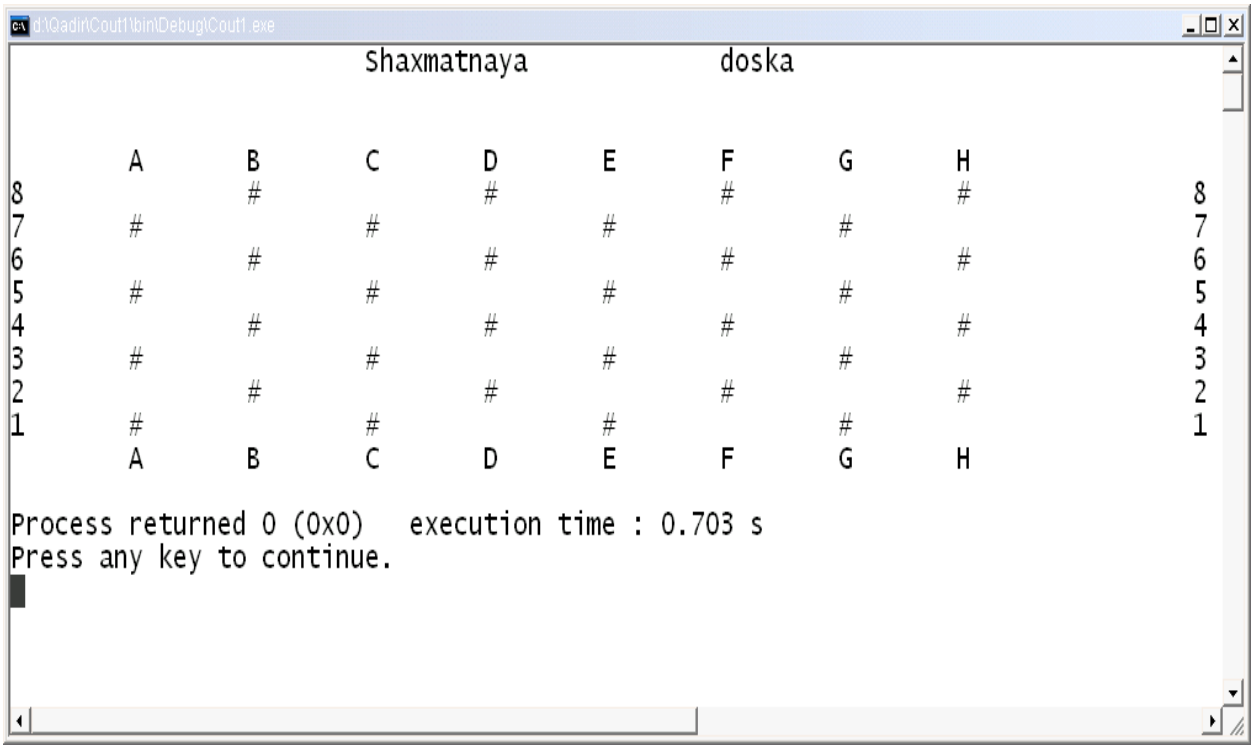
При истинном условии в операторе if оператор continue вызывает пропуск идущих за ним операторов тела цикла и осуществляется переход к началу следующей итерации.

Практический часть

1-//Shaxmatnaya doska

```
#include <iostream>

using namespace std;

int main()
{
cout << "\t\t\tShaxmatnaya\t\tdoska\n\n\n";

cout << "\t";
for (char simvol='A'; simvol<='H'; simvol++)
cout << simvol << "\t";

cout << endl;

for (int i=8; i>=1; i--)
{
cout <<i<<"\t";
if (i%2!=0)
cout <<"#\t\t#\t\t#\t\t#\t\t\t" <<i<<"\n";
else
cout <<"\t#\t\t#\t\t#\t\t#\t\t" <<i<<"\n";
}

cout << "\t";
for (char simvol='A'; simvol<='H'; simvol++)
cout << simvol << "\t";
```

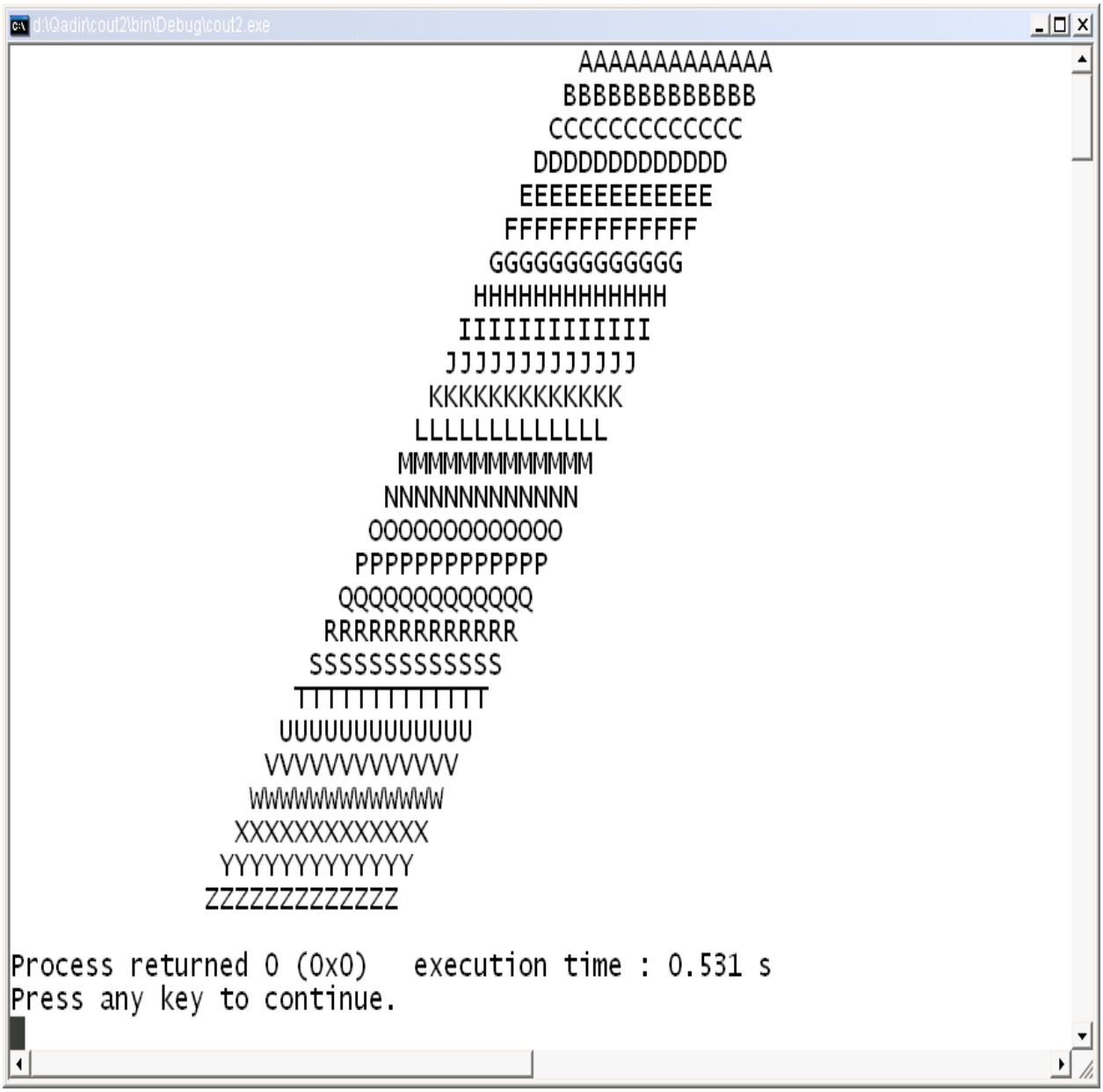
```
cout << endl;


return 0;
}
```



2-

```
//Изображение диагонали, состоящей из букв алфавита;
#include <iostream>


using namespace std;


int main()
{
int j=38;
for (char i='A'; i<='Z'; i++)
{
for (int h=1; h<=j; h++)
cout << " ";
cout <<i<<i<<i<<i<<i<<i<<i<<i<<i<<i<<i<<i<<endl;
```

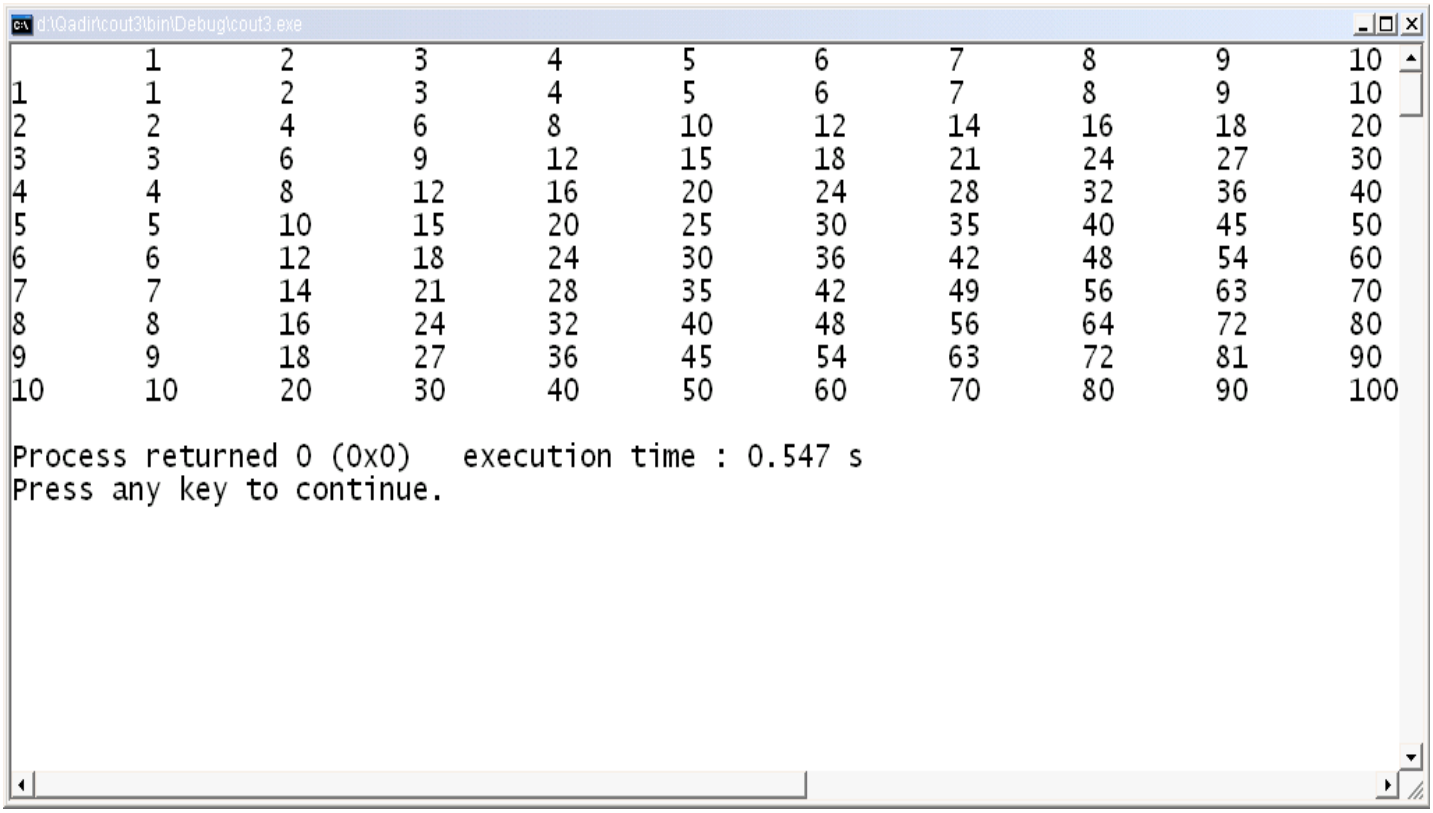
```
j--;
}
return 0;
}
```



3.Tablica umnajenie

```
#include <iostream>

using namespace std;

int main()
{
```

```
cout << "\t";
for (int i=1; i<=10; i++)
{
cout << i << "\t";
}
cout<<endl;

for (int i=1; i<=10; i++)
{
cout << i <<"\t";
for (int j=1; j<=10; j++)
{
cout << i*j << "\t";
}
cout << endl;
}
return 0;
}
```

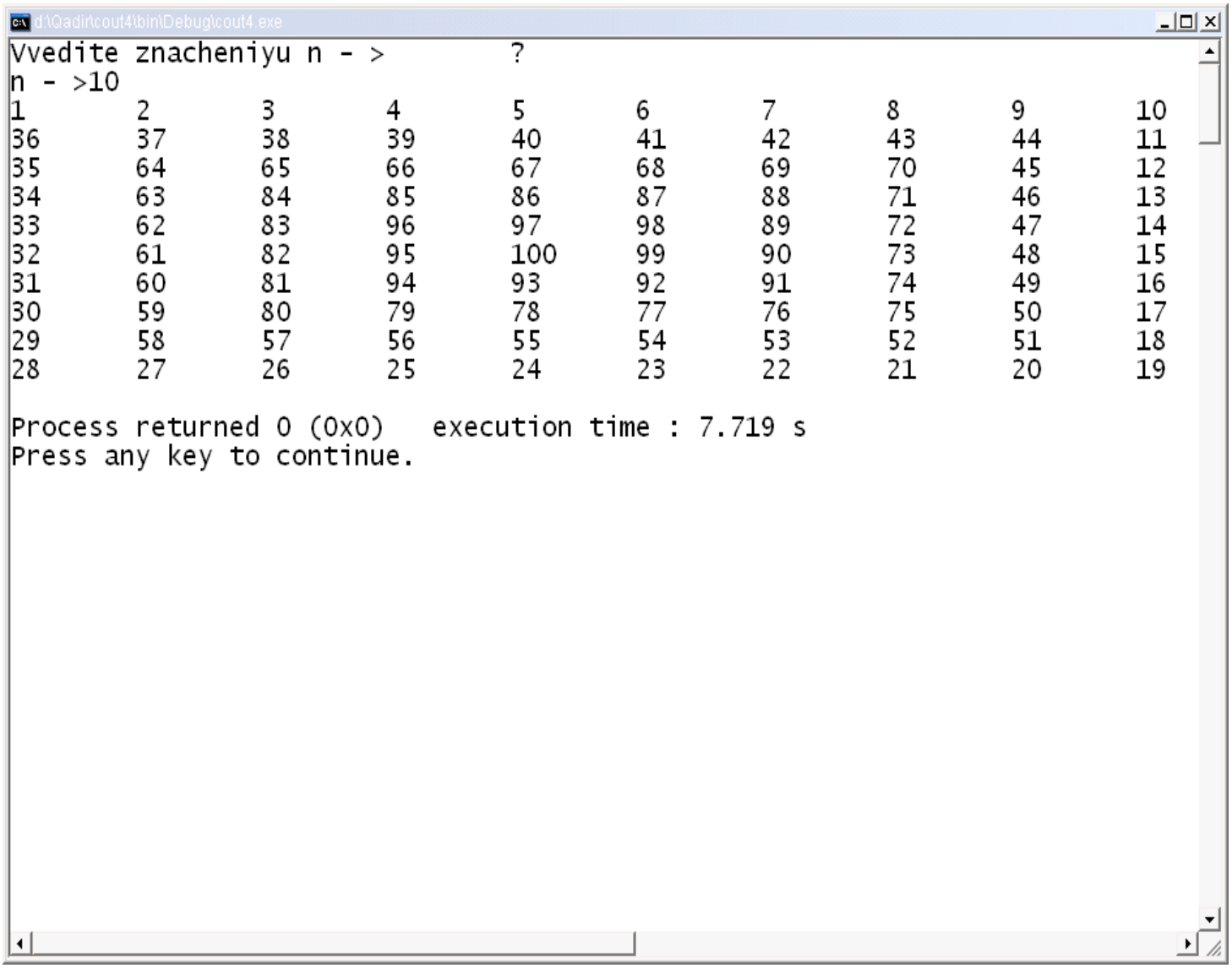


4-

```
//Primer dlya operatoru cin [vvoda]
//Etot zadach iz sayta acm.tuit.uz [Spiral]
#include <iostream>
using namespace std;

int main()
{
long int n;
cout << "Vvedite znacheniyu n - >\t?\n";
cout << "n - >";    cin>>n;
int a[n][n];
int k=1,k1=0,k2=n;
```

```
for (int k=1; k<=n*n;)
{
for (int i=k1; i<k2; i++)
{
a[k1][i]=k;
k=k+1;
}
for (int i=k1+1; i<k2; i++)
{
a[i][k2-1]=k;
k=k+1;
}
for (int i=k2-2; i>=k1; i--)
{
a[k2-1][i]=k;
k=k+1;
}
for (int i=k2-2; i>=k1+1; i--)
{
a[i][k1]=k;
k=k+1;
}
k2=k2-1;
k1=k1+1;
}
for (int i=0; i<n; i++)
{
for (int j=0; j<n; j++)
cout<<a[i][j]<<"\t";
cout<<"\n";
```

```
}
return 0;
}
```


d:\Qadir\cout4\bin\Debug\cout4.exe
Vvedite znacheniyu n - >   ?
n - >10
1   2   3   4   5   6   7   8   9   10
36   37   38   39   40   41   42   43   44   11
35   64   65   66   67   68   69   70   45   12
34   63   84   85   86   87   88   71   46   13
33   62   83   96   97   98   89   72   47   14
32   61   82   95   100   99   90   73   48   15
31   60   81   94   93   92   91   74   49   16
30   59   80   79   78   77   76   75   50   17
29   58   57   56   55   54   53   52   51   18
28   27   26   25   24   23   22   21   20   19
Process returned 0 (0x0)   execution time : 7.719 s
Press any key to continue.

5-O'ZBEKISTON RESPUBLIKASINING DAVLAT MADHIYASI:

```
#include <iostream>

using namespace std;

int main()
{
cout << "\t\tO'ZBEKISTON RESPUBLIKASINING\n\t\t      DAVLAT MADHIYASI\n\n";
cout << "\t\t\t\t\t\tMUTAL BURHONOV musiqasi\n";
cout << "\t\t\t\t\t\tABDULLA ORIPOV so'zi\n\n\n";

cout << "\tSerquyosh, hur o'lkam, elga baxt najot,\n";
cout << "\tSen o'zing do'stlarga yo'ldosh, mehribon !\n";
cout << "\tYahnagay to abad ilmu fan, ijod,\n";
cout << "\tShuhrating porlasin toki bor jahon !\n\n\n";

cout << "\t Naqorat:\n";

cout << "\tOltin bu vodiylar-jon O'zbekiston,\n";
cout << "\tAjdodlar mardona ruhi senga yor !\n";
cout << "\tUlug' xalq qudrati jo'sh urgan zamon,\n";
cout << "\tOlamni mahliyo aylagan diyor !\n\n\n";

cout << "\tBag'ri keng o'zbekning o'chmas iymoni,\n";
cout << "\tErkin, yosh avlodlar senga zo'r qanot,\n";
cout << "\tIstiqlol masha'li, tinchlik posboni,\n";
cout << "\tHaqsevar ona yurt, mangu bo'l obod !\n\n\n";

cout << "\t Naqorat:\n";
```

```
cout << "\tOltin bu vodiylar-jon O'zbekiston,\n";
cout << "\tAjdodlar mardona ruhi senga yor !\n";
cout << "\tUlug' xalq qudrati jo'sh urgan zamon,\n";
cout << "\tOlamni mahliyo aylagan diyor !\n\n\n";

return 0;
}
```

d:\Qadir\cout5\bin\Debug\cout5.exe

```
                O'ZBEKISTON RESPUBLIKASINING
                      DAVLAT MADHIYASI

                                                MUTAL BURHONOV musiqasi
                                                ABDULLA ORIPOV so'zi


        Serquyosh, hur o'lkam, elga baxt najot,
        Sen o'zing do'stlarga yo'ldosh, mehribon !
        Yahnagay to abad ilmu fan, ijod,
        Shuhrating porlasin toki bor jahon !


         Naqorat:
        Oltin bu vodiylar-jon O'zbekiston,
        Ajdodlar mardona ruhi senga yor !
        Ulug' xalq qudrati jo'sh urgan zamon,
        Olamni mahliyo aylagan diyor !


        Bag'ri keng o'zbekning o'chmas iymoni,
        Erkin, yosh avlodlar senga zo'r qanot,
        Istiqlol masha'li, tinchlik posboni,
        Haqsevar ona yurt, mangu bo'l obod !


         Naqorat:
        Oltin bu vodiylar-jon O'zbekiston,
        Ajdodlar mardona ruhi senga yor !
        Ulug' xalq qudrati jo'sh urgan zamon,
        Olamni mahliyo aylagan diyor !



Process returned 0 (0x0)   execution time : 0.203 s
Press any key to continue.
```

Литература

1. Подбельской В.В. Язык С++: Учебное пособие. - М.: Финансы и статистика,1995, - 560 с.
2. Страуструп Б. Язык программирования С++. - М.: Радио и связь, 1991. - 352 стр.
3. Романов В.Ю. Программирование на языке С++. Практический подход. - М.: Компьтер, 1993. - 160 с.
4. Юлин В.А., Булатова И.Р. Приглашение к С++. - Мн.: Высш. Шк., 1990,- 224 с.
5. Котлинская Г.П., Галиновский О.И. Программирование на языке С++. -Мн.: Высш. Шк., 1991. - 156 с.
6. www.allbest.ru
7. www.referat.uz
8. www.Ziyonet.uz