O’ZBEKISTON RESPUBLIKASI ALOQA, AXBOROTLASHTIRISH VA TELEKOMUNIKATSIYA DAVLAT QO’MITASI
TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

KURS ISHI

Mavzu: Konteynerlar

Bajardi: Buriboyev Shahboz_____

Qabul qildi: _____

Toshkent-2013

Mundarija:

Kirish

Hozirgi kunda ma’lumotlar ustida ishlash kundan kunga kattalashib bormoqda, shuning uchun yangi qo’shimcha ma’lumot toifalarini yaratsih ulardan foydalanish asosiy masalalardan biri hisoblanadi.

Dasturlash tillarining kutubxonalari bilan kengroq tanishish ularning sintaksisini o’rganib, dinamik ma’lumotlar toifasini yaratgan holda ular ustida amallar bajarish.

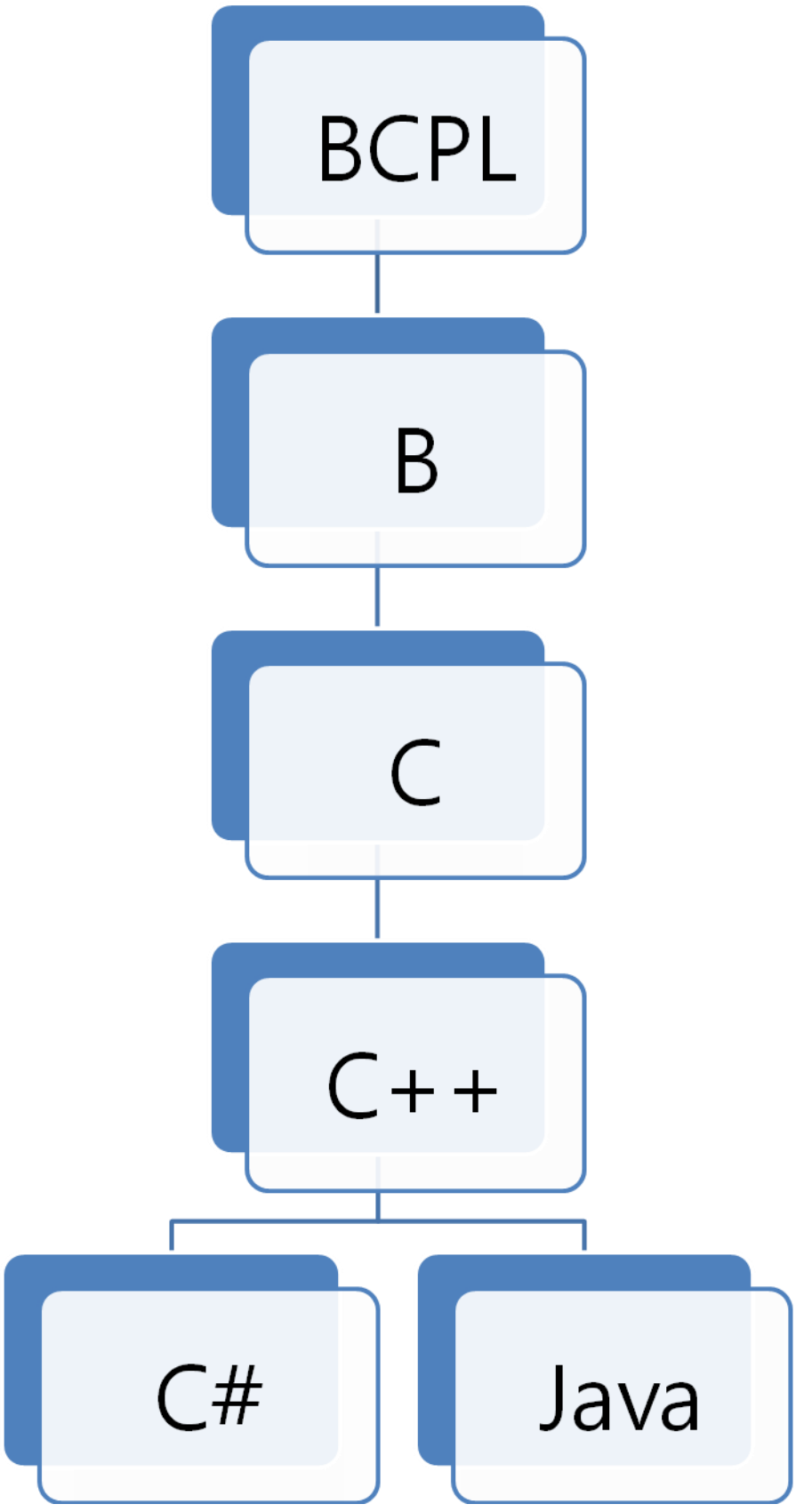
C++ dasturlsh tili imkoniyatlaridan keng holda foydalangan holda uning kutubxonalariga murojat qilish ularni ishlash prinsiplarini o’rganish va ular asosida xususiy konteynerlar yaratish.

C++ DASTURLASH TILINING KELIB CHIQISHI XAQIDA MA'LUMOT

C++ dasturlash tili C tiliga asoslangan. C esa o'z navbatida B va BCPL tillaridan kelib chiqqan. BCPL 1967 yilda Martin Richards tomonidan tuzilgan va operatsion sistemalarni yozish uchun mo'ljallangan edi. Ken Thompson o'zining B tilida BCPL ning ko'p hossalarini kiritgan va B da UNIX operatsion sistemasining birinchi versiyalarini yozgan. BCPL ham, B ham tipsiz til bo'lgan. Yani o'garuvchilarning ma'lum bir tipi bo'lmagan - har bir o'zgaruvchi kompyuter hotirasida faqat bir bayt yer egallagan. O'zgaruvchini qanday sifatda ishlatish esa, yani butun sonmi, kasrli sonmi yoki harfdekmi, dasturchi vazifasi bo'lgan. C tilini Dennis Ritchie B dan keltirib chiqardi va uni 1972 yili ilk bor Bell Laboratoriyasida, DEC PDP-11 kompyuterida qo'lladi. C o'zidan oldingi B va BCPL tillarining juda ko'p muhim tomonlarini o'z ichiga olish bilan bir qatorda o'zgaruvchilarni tiplashtirdi va bir qator boshqa yangiliklarni kiritdi. Boshlanishda C asosan UNIX sistemalarida keng tarqaldi. Hozirda operatsion sistemalarning asosiy qismi C/C++ da yozilmoqda. C mashina arhitekturasiga bog'langan tildir. Lekin yahshi rejalashtirish orqali dasturlarni turli kompyuter platformalarida ishlaydigan qilsa bo'ladi. 1983 yilda, C tili keng tarqalganligi sababli, uni standartlash harakati boshlandi. Buning uchun Amerika Milliy Standartlar Komiteti (ANSI) qoshida X3J11 tehnik komitet tuzildi. Va 1989 yilda ushbu standart qabul qilindi. Standartni dunyo bo'yicha keng tarqatish maqsadida 1990 yilda ANSI va Dunyo Standartlar Tashkiloti (ISO) hamkorlikda C ning ANSI/ISO 9899:1990 standartini qabul qilishdi.

Shu sababli C da yozilgan dasturlar kam miqdordagi o'zgarishlar yoki umuman o'zgarishlarsiz juda ko'p kompyuter platformalarida ishlaydi. C++ 1980 yillar boshida Bjarne Stroustrup tomonidan C ga asoslangan tarzda tuzildi. C++ juda ko'p qo'shimchalarni o'z ichiga olgan, lekin eng asosiysi u ob'ektlar bilan dasturlashga imkon beradi. Dasturlarni tez va sifatli yozish hozirgi kunda

katta ahamiyat kasb etmoda. Buni ta'minlash uchun ob'ektli dasturlash g'oyasi ilgari surildi. Huddi 70-chi yillar boshida strukturali dasturlash kabi, programmalarni hayotdagi jismlarni modellashtiruvchi ob'ektlat orqali tuzish dasturlash sohasida inqilob qildi. C++ dan tashqari boshqa ko'p ob'ektli dasturlshga yo'naltirilgan tillar paydo bo'ldi. Shulardan eng ko'zga tashlanadigani Xerox ning Palo Altoda joylashgan ilmiy-qidiruv markazida (PARC) tuzilgan Smalltalk dasturlash tilidir. Smalltalk da hamma narsa ob'ektlarga asoslangan. C++ esa gibrid tildir. Unda C ga o'hshab strukturali dasturlash yoki yangicha, ob'ektlar bilan dasturlash mumkin. Yangicha deyishimiz ham nisbiydir. Ob'ektli dasturlash falsafasi paydo bo'lganiga ham yigirma yildan oshayapti. C++ funksiya va ob'ektlarning juda boy kutubhonasiga ega. Yani C++ da dasturlashni o'rganish ikki qismga bo'linadi. Birinchisi bu C++ ni o'zini o'rganish, ikkinchisi esa C++ ning standart kutubhonasidagi tayyor ob'ekt/funksiyalarni qo'llashni o'rganishdir.



1-rasm Dasturlash tillarining rivojlanishi

C++ dasturlash tilida STL bilan tanishish

Umumlashgan yoki unifikasiyalangan dasturlashning maqsadi tartiblash kabi ko‘p qo‘llaniluvchi algoritmlar va sinflar saqlanuvchi universal kutubxonalar yaratish orqali dasturlash jarayonini avtomatlashtirishdan iboratdir. Shu bilan birga, bu kutubxonaga kiruvchi funksiyalar universal xarakterga ega bo‘lishi, ya’ni ixtiyoriy turdagi ma’lumotlar ustida amallar bajarish imkonini berishi lozim.

Shablonlarga asoslangan umumlashgan dasturlashga misol Stepanov va Target tomonidan yaratilgan va C++ tili standartiga kiritilgan STL (Standart Template Library) kutubxonasidir. Kutubxona yadrosi uchta elementdan iborat:

- Konteynerlar,
- Algoritmlar
- Iteratorlar.

Konteynerlar — bu boshqa elementlarni saqlash uchun mo‘ljallangan sinflar shablonlaridir. Konteynerlar asosiy xususiyati shundaki ular ixtiyoriy tipdagi elementlarni o‘zida saqlash uchun mo‘ljallangan. To‘g‘rirog‘i, har bir tur uchun shablon nusxasi kerak bo‘lganda, kompilyator tomonidan avtomatik tarzda yaratiladi. Algoritmlar konteyner elementlari ustidan operasiyalar bajaradi. Bibliotekada qidirish, saralash va almashtirish uchun algoritmlar mavjud. Algoritmlar elementlar ketma_ketligi bilan ishlash uchun mo‘ljallangan. Algoritmlar asosiy xususiyati shuki ular ixtiyoriy konteynerlar bilan ishlay oladi.

Konteynerlar asosiy va hosila konteynerlarga ajratiladi. Asosiy konteynerlarga quyidagilar kiradi:

- **vector** — dinamik massiv
- **list** — chiziqli ro‘yxat
- **deque** — ikki tarafli tartib
- **set** — to‘plam

• **multiset** — har bir elementi noyob bo‘lishi shart emas to‘plam

• **map** — kalit/ qiymat juftlikni saqlash uchun assosiativ ro‘yxat. Bunda har bir kalit bitta qiymat bilan bog‘langan.

• **multimap** — har bir kalit bilan ikkita yoki ko‘proq qiymatlarbog‘langan

Hosila konteynerlarga quyidagilar kiradi:

• **stack** — stek

• **queue** — tartib

• **priority_queue** — prioritetli tartib

STL kutubxonasidagi standart shablonlardan foydalanish uchun kerakli header fayllarni dasturga ulash lozim.

vector

Birinchi bo’lib STL dagi vector bilan ishlaymiz. Buning uchun vector header faylini dasturga ulaymiz.

Vector tipidagi o’zgaruvchi yaratamiz. Buning uchun

vector <type> var_name

Bu yerda

- type – vector tarkikibiga kiruvchi o’zgaruvchilarning toifasi
- var_name – vectorning nomi

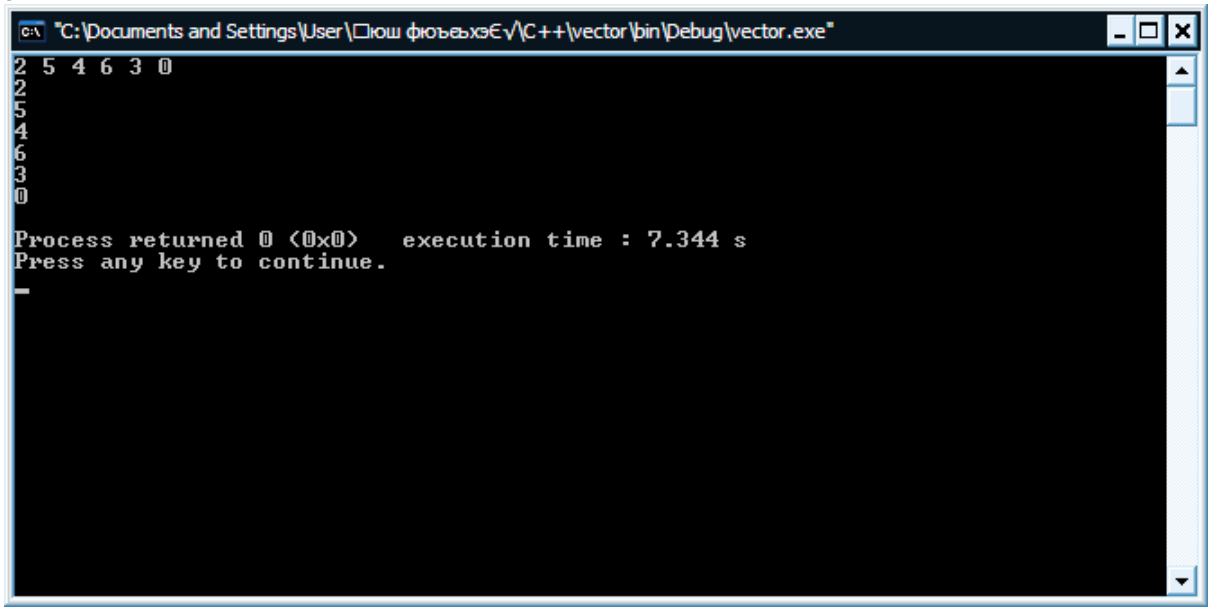
STL kutubxonasidagi maxsus vectorning ichiga ma’lumot qo’shish uchun quyidagi funksiyadan foydalaniladi.

push_back(value)

- value –vectorga qo’shiluvchi qiymat

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector <int> vc; // vectorni e’lon qilish
    int a;
    cin>>a;
```

```
    vc.push_back(a);
    while(a)
    {
        cin>>a;
    vc.push_back(a);
      }

      for(int i=0;i<vc.size();i++)
    cout << vc[i] << endl;
    return 0;
}
```



Ro’yxat

STL kutubxonasidagi list konteyneri bilan ishlash. Buning uchun eng avvalo list header faylini dasturimizga ulaymiz.

List tipidagi o’zgaruvchini yaratish:

list <type> list_name;

STL kutubxonasidagi maxsus vectorning ichiga ma’lumot qo’shish uchun quyidagi funksiyalardan foydalaniladi.

push_back(value) – listning oxiriga qo’shish

push_front(value) – listning boshiga qo’shish

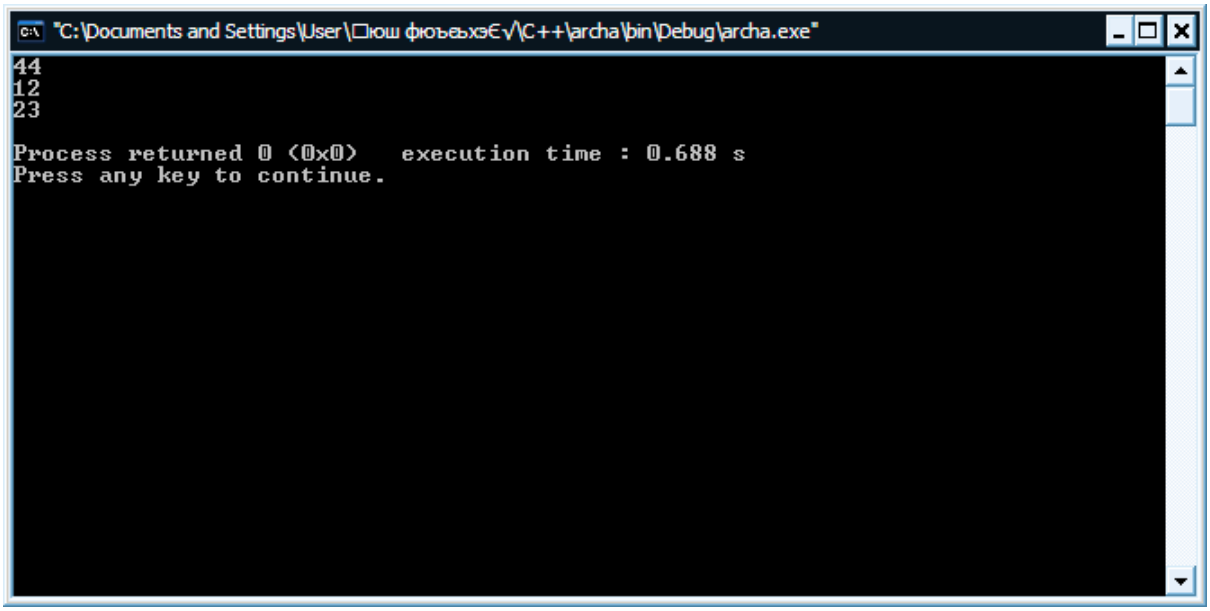
List elementlariga murojatni amalga oshirish uchun iteratorlardan foydalanish zarur.

Iteratorlar — bu konteyner hamma elementlarini ko‘rib chiqish va qayta ishlashga imkon beruvchi obyektlardir. Iteratorlar algoritmlar universalligini ta’minlovchi asosiy vositadir.

Iteratorlardan foydalanish uchun ma’lum list konteyneriga most iteratorlar yaratish lozim.

list<type>::iterator iterator_name

```
#include <iostream>
#include <list>
using namespace std;
int main()
{
	list <int> lst;
	lst.push_back(12);
	lst.push_back(23);
	lst.push_front(44);
	list<int>::iterator it;
	for(it=lst.begin();it!=lst.end();it++)
	cout<<*it<<endl;
	return 0;
}
```



Stack

LIFO (Last in first out) ya'ni navbatning oxirgi bo’lib kirgan elementiga birinchi bo’lib xizmat ko’rsatiladi. Bu eng ko’p ishlatiladigan ma'lumotlar

tuzilmalaridan biri bo’lib, turli xil masalalarni hal qilishda ancha qulay va samarali xisoblanadi. Xizmat ko’rsatishni keltirilgan tartibiga ko’ra, stackda faqatgina bitta pozitsiyaga murojaat qilish mumkin. Bu pozitsiya stackning uchi deyilib unda stackka vaqt bo’yicha eng oxirgi kelib tushgan element nazarda tutiladi. Biz stackga yangi element kiritsak, bu element oldingi stack uchida turgan element ustiga joylashtiriladi xamda stackni uchida joylashib qoladi. Elementnifaqatgina stack uchidan tanlash mumkin; bunda tanlangan element stackdan chiqarib tashlanadi va stack uchini esa chiqarib tashlangan elementdan bitta oldin kelib tushgan element tashkil qilib qoladi. (bunday tuzilmaga ma'lumotlarga cheklangan murojaat tuzilmasi deyiladi).

Stackni grafik ko’rinishida quyidagicha tasvirlash mumkin:

2-rasm Stack

Stack ko’rinishidagi konteynerlar bilan ishlash. Buning uchun stack header faylini dasturga ulash lozim.

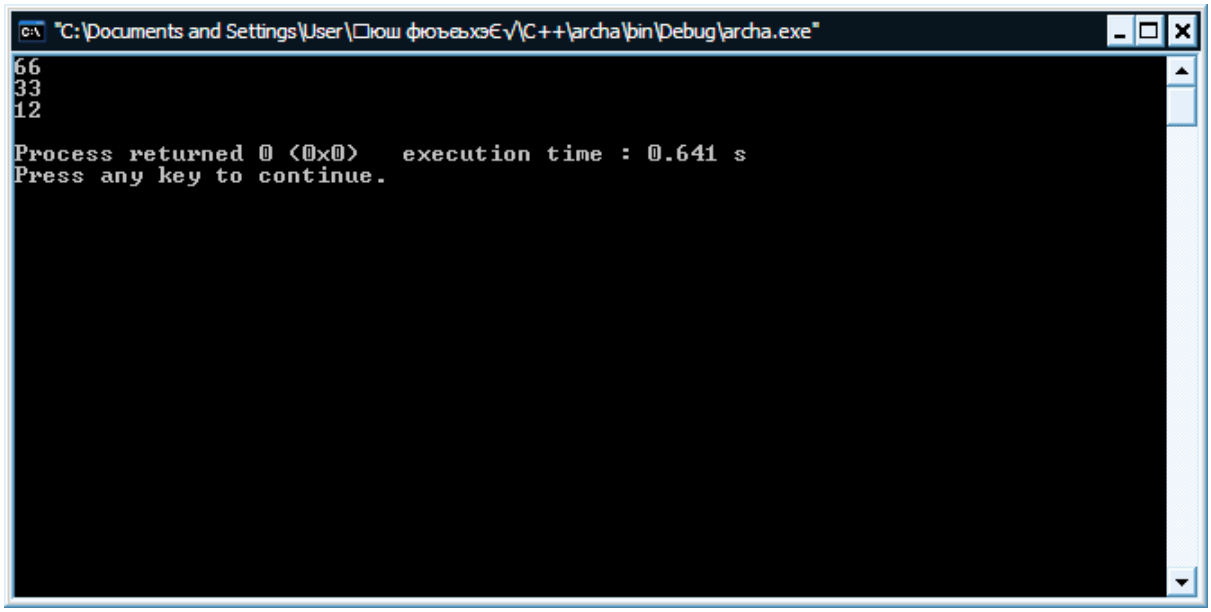
Stack ustida amalga oshiriladigan amallar:

1. PUSH(i) - stackga element kiritish, i - stackga kiritiladigan element;
2. POP ( ) - stackdan elementni tanlash. Element tanlanayotganda o’zi egallab turgan ishchi xotiraga joylashtiriladi;
3. EMPTY ( ) - stackni bo’sh yoki bo’sh emasligini tekshirish (true - bo’sh, false bo’sh emas);
4. TOP ( ) - stack yuqori elementini o’chirmasdan o’qish.

Stack tipidagi o’zgaruvchini quydagicha e’lon qilishimiz lozim.

stack <type> stack_name;

```
#include <iostream>
#include <stack>
using namespace std;
int main()
{
    stack <int> sc;
    sc.push(12);
    sc.push(33);
    sc.push(66);

    while(!sc.empty())
    {
        cout<<sc.top()<<endl;
        sc.pop();
}
}
```

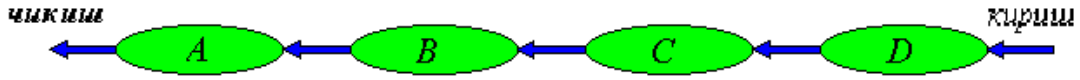


Navbat

Dasturlashda shunday ma'lumotlar tuzilmasi mavjudki, u navbat deyiladi.

Bunday ma'lumotlar tuzilmasi real navbatni modellashtirishda katta axamiyatga

ega. Bunda xizmat ko’rsatishga kelib tushgan talab, uning ijrosi, ya'ni xizmat ko’rsatish tartibini aniqlashda zarur bo’ladi. Kundalik qayotimizdan barchamizga ma'lum bo’lgan navbat turi, dasturlashda FIFO (First input-First output, ya'ni birinchi kelgan - birinchi ketadi) deb nomlanadi. quyida 4 ta elementdan iborat navbat keltirilgan.



Bu erdan ko’rinib turibdiki, stekdan farqli ravishda xizmat ko’rsatilish birinchi kelgan elementga birinchi bo’lib xizmat ko’rsatiladi. Stekdan yana bir farqi, bunda navbatning har ikkala tomoni ochiq bo’ladi, ya'ni bir tomondan kelib ikkinchi tomondan chiqib ketadi.

Demak, navbatda elementni olish ro’yxat boshidan, yozish esa oxiridan amalga oshiriladi.

EXM xotirasida real navbat eementlari soni chekli bo’lgan bir o’lchamli massiv ko’rinishida yaratiladi. Albatta, bunda navbat elementi turini ko’rsatish va navbat bilan ishlashni ko’rsatuvchi o’zgaruvchi zarur bo’ladi.

Navbat fizik bosqichda xotira sohasini ro’yxat ketma-ketligi bo’yicha to’laligicha egallaydi.

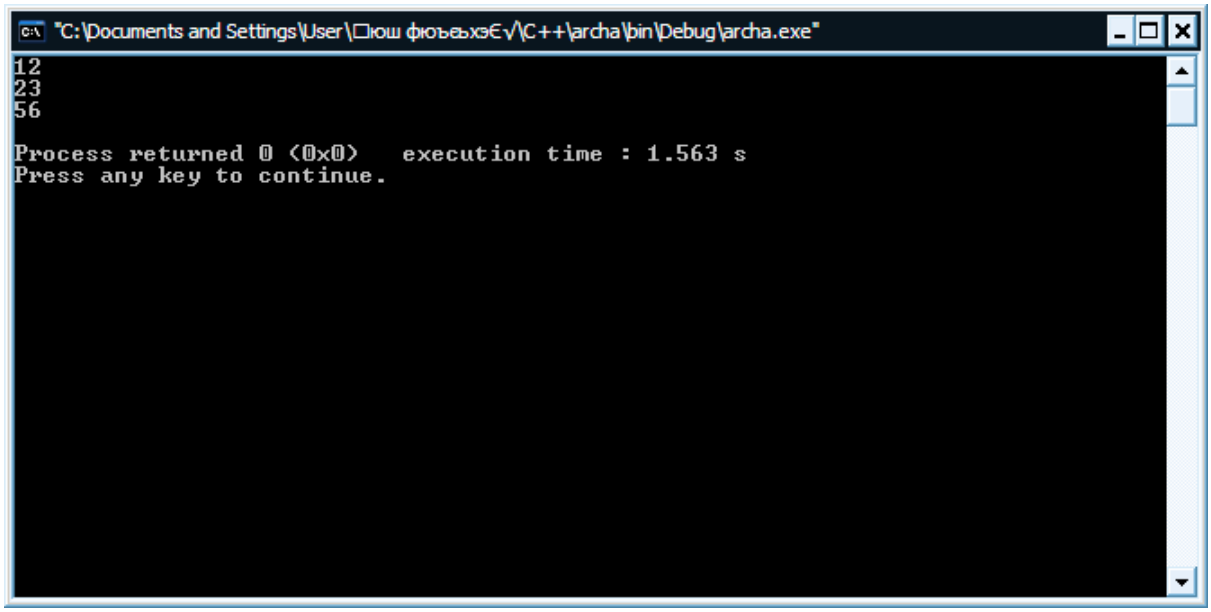
Navbat ustida amalga oshiriladigan amallar:

Navbat uchun 3 ta oddiy amal aniqlangan.

1. Navbatga yangi element joylashtirish: insert (x), x - element.
2. Navbat boshidan elementni o’chirish: remove()
3. Navbatni bo’sh yoki bo’sh emasligini aniqlash: empty ()
4. Navbat elementlariga murojatni ta’minlashda foydalaniladi: front ()

```
#include <iostream>
#inclussssde <queue>

using namespace std;
```

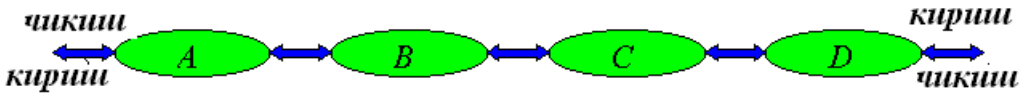
```
int main()
{
   queue <int> qu;
    qu.push(12);
    qu.push(23);
    qu.push(56);
    while(!qu.empty())
    {
          cout<<qu.front()<<endl;
          qu.pop();
}
}
```



Dek

Dek so’zi (DEQ - Double Ended Queue) ingliz tilidan olingan bo’lib, ikkita chetga ega navbat degan ma'noni bildiradi.

Dekning o’ziga xos xususiyati shundan iboratki, elementlarni yozish va o’qishni har ikkala chetidan xam amalga oshirish mumkin.

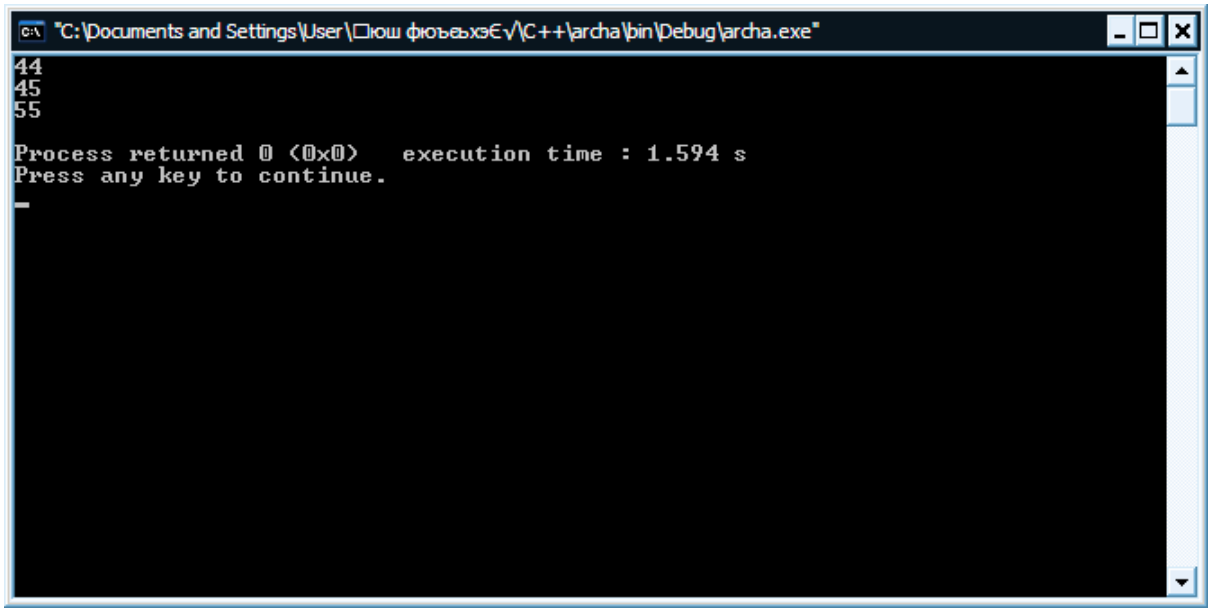


Dekni quyi chegaralari birlashtirilgan ikkita stek ko’rinishda qarash

mumkin. Dek ustida bajariladigan amallar:

- Insert - element qo’yish.
- Remove - dekdan elementni chiqarib tashlash.
- Empty - bo’sh yoki bo’sh emasligini tekshirish.
- Full - to’lalikka tekshirish.

```
#include <iostream>
#include <deque>
using namespace std;
int main()
{
   deque<int> dq;
   dq.push_back(45);
   dq.push_back(55);
   dq.push_front(44);
   deque<int>::iterator it;
   for(it=dq.begin();it!=dq.end();it++)
   cout<<*it<<endl;
}
```



C++ dasturlash tilida xususiy konteynerlar yaratish

Ma’lumotlar konteynerlarini yaratishda C++ tilining C tilidan meros qilib olgan structuralardan foydalaniladi. Dasturlash tillaridagi imkoniyatlari ichida C/C++ tillining ko’rsatkichlar bilan ishlash imkoniyati yuqori hisoblanadi shuning uchun biz ma’lum konteynerlarni o’zimiz xotiraga bevosita murojat qilish orqali yaratishimiz mumkin.

Ma’lumotlar konteynerlarini yaratishda quyidagi konteynerlarni ko’rib chiqamiz.

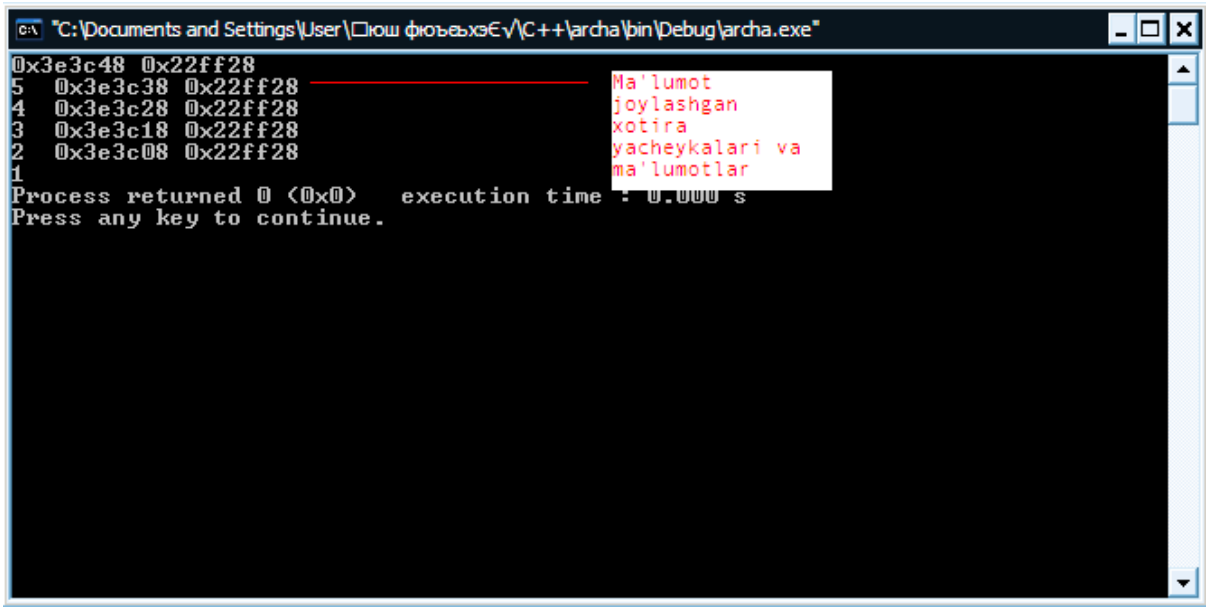
- Stack
- Navbat
- Ro’yxat
- Binar daraxt (Binary tree)

Dastur yechimi

STACK ko’rinishidagi konteyner

```
#include <iostream>
#include<cstdio>
#include<cstring>
#include<cstdlib>

using namespace std;

struct Node//stack uchun kontener
{
int info;
Node *pointer;
};
Node *first(int d);//birinchi elementni qo'shish
```

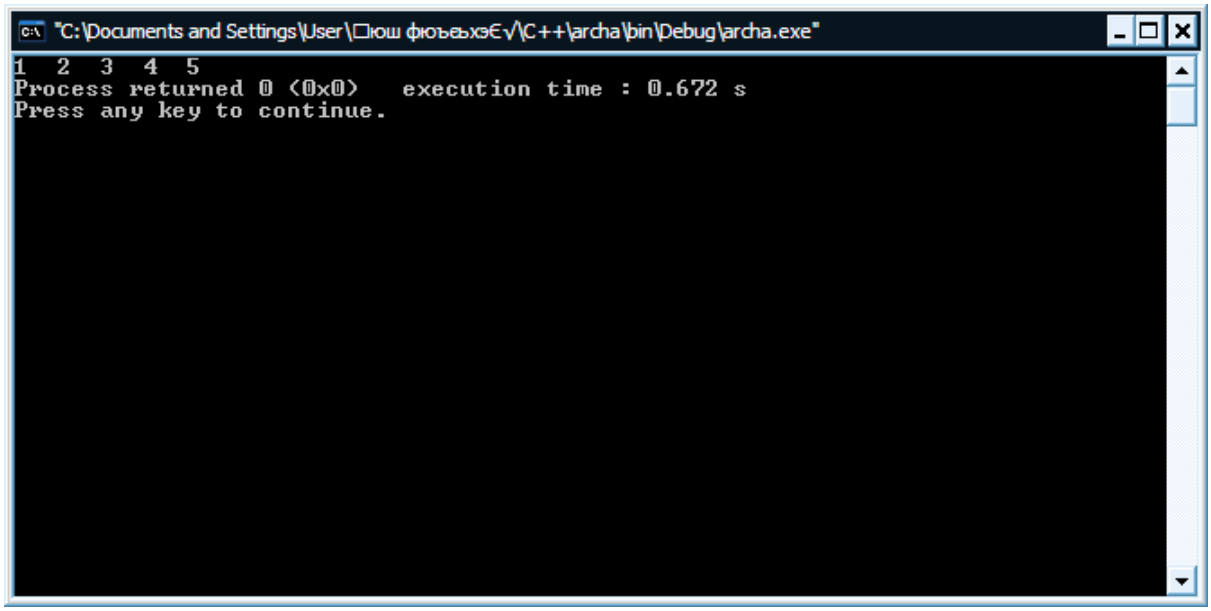
```
        void push(Node **top, int d);//yangi element qo'shish
        int pop(Node **top);//elementni o'chirish

        int main()
        {
            Node *top =first(1);
            for(int i=2;i<6;i++) push(&top,i);
            while(top)
            {     cout<<top<<" "<<&top<<endl;
                  cout<<pop(&top)<<"    ";
            }
            return 0;
        }
        Node *first(int d)
        {
            Node *pv=new Node;//yangi kontener yaratish
            pv->info=d;//yangi kontenerni ma'lumot yacheykasiga d ma'lumotni
        qo'yamiz
            pv->pointer=0;//keyingi element hali yo'q nol gs tenglaymiz
            return pv;//kontener addressini qaytaramiz
        }
        void push(Node **top, int d)
        {
Node *pv=new Node;//yangi kontener yaratish
pv->info=d;//yangi kontenerni ma'lumot yacheykasiga d ma'lumotni qo'yamiz
pv->pointer=*top;////yangi kontenerni keyingi oldigi kontener bilan bog'laymiz
            *top=pv;//stack boshiga yangi elemntni qo'yamiz
        }
        int pop(Node **top)
        {
            int temp=(*top)->info;//kontenerdagi ma'lumotni tempga olamiz
            Node *pv=*top;//yangi kontenerga stack boshini beramiz
            *top=(*top)->pointer;//stack boshini keyingi elementga o'tkazib
            delete pv;//dinamik ajratilgan joyni o'chiramiz
            return temp;//ma'lumotni qaytaramiz
}
```



Navbat ko’rinishidagi konteyner

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

using namespace std;

struct Node//navbat uchun kontener
{
	int d;
	Node *pointer;
};
Node* first(int d);//birinchi elementni qo'shish
void add(Node **pend, int d);//yangi element qo'shish
int remove_items(Node **pbegin);//elementni o'chirish

int main()
{
Node *pbeg=first(1);
Node *pend=pbeg;
printf("%p\n",pbeg);
for(int i=2;i<6;i++) add(&pend,i);
while(pbeg)
	cout<<remove_items(&pbeg)<<"    ";
return 0;
}
Node *first(int d)
```

```
{
	Node *pv=new Node;//yangi kontener yaratish
	pv->d=d;//yangi kontenerni ma'lumot yacheykasiga d ma'lumotni qo'yamiz
	pv->pointer=0;//keyingi element hali yo'q nol gs tenglaymiz
	return pv;//kontener addressini qaytaramiz
}
void add(Node **pend,int d)
{
	Node *pv=new Node;//yangi kontener yaratish
	pv->d=d;//yangi kontenerni ma'lumot yacheykasiga d ma'lumotni qo'yamiz
	pv->pointer=0;//keyingi element hali yo'q nol gs tenglaymiz
    (*pend)->pointer=pv;//oldingi elementni keyingisiga yangi kontenerni ko'rsatamiz
	*pend=pv;//oxirgi element yangi kontener bo'ladi
}

int remove_items(Node **pbegin)
{
	int temp=(*pbegin)->d;//kontenerdagi ma'lumotni tempga olamiz
	Node *pv=*pbegin;//yangi kontenerga navbat boshini beramiz
	*pbegin=(*pbegin)->pointer;//navbat boshini keyingi elementga o'tkazib
	delete pv;//dinamik ajratilgan joyni o'chiramiz
	return temp;//ma'lumotni qaytaramiz
}
```



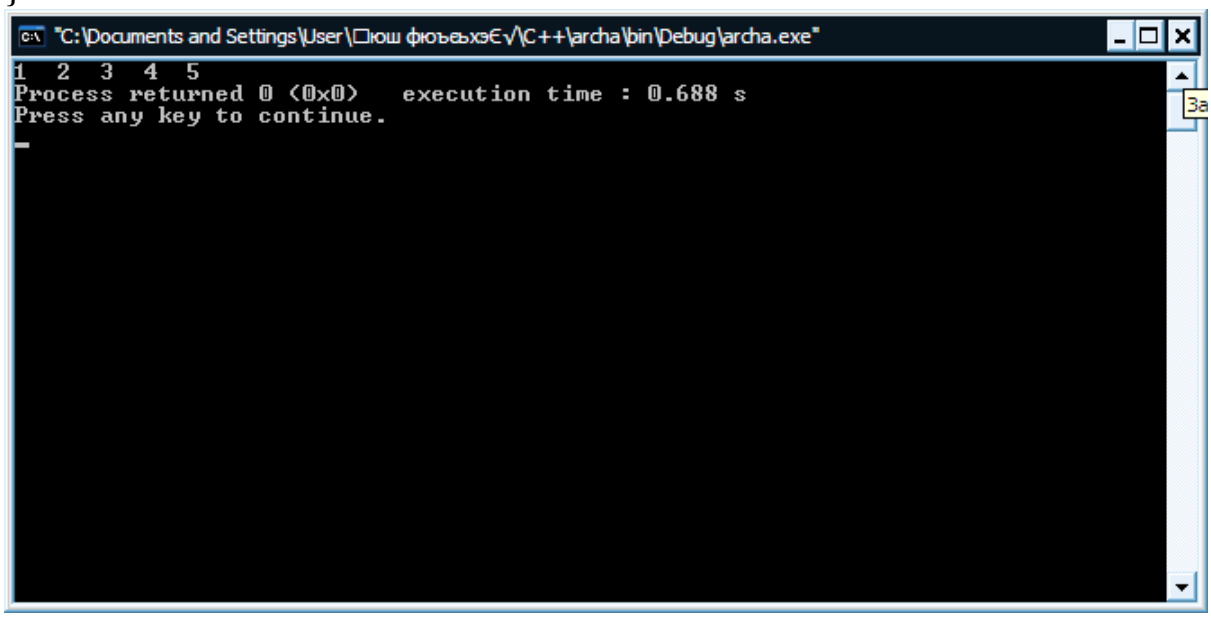
Ro’yxat ko’rinishidagi konteyner

```
#include <iostream>
```

```
// Ro'yxat
using namespace std;
struct Node//Ma'lumot saqlovchi kontener
{
int data;//saqlanadigan ma'lumot
Node *prev;//oldingi elementga ko'rsatkich
Node *next;//keyingi elementga ko'rsatkich
};

Node* first(int d);//ro'yxatni birinchi elementini qo'shish
void add(Node **pend, int d);//yangi element qo'shish
Node *finder(Node *const pbeg,int key);//ro'yxatdan kalit bo'yicha qidirish
bool removed(Node **pbeg,Node **pend,int key);//berilgan kalit bo'yicha o'chirish
Node *inserted(Node *const pbeg, Node **pend,int key,int d);//berilgan kalitdan keyinga qo'shish
int main()
{
    Node *pbeg=first(1);//ro'yxatni birinchi elementini qo'shib uning addresini olish
    Node *pend=pbeg;//birinchi element qo'shilganda oxirgisi ham o'zi bo'ladi
    for(int i=2;i<6;i++) add(&pend,i);//yangi element qo'shish
    Node *pv=pbeg;
    while(pv)//ro'yxat elementlarini chiqarish
      {
    cout<<pv->data<< "    ";
    pv=pv->next;
      }
}

Node* first(int d)
{
      Node *pv=new Node;//yangi kontener yaratish
      pv->data=d;//kontenerni data yacheykasiga d ma'lumotni qo'shish
      pv->next=0;//birinchi element uchun keyingi element bo'lmaydi
      pv->prev=0;//birinchi element uchun oldingi element bo'lmaydi
      return pv;// kotener addresini qaytarish
}
void add(Node **pend, int d)
{
      Node *pv =new Node;//yangi kontener yaratish
      pv->data=d;//yangi kontenerni ma'lumot yacheykasiga d ma'lumotni qo'yamiz
      pv->next=0;//kontenerni keyingi kontenerga ko'rsatkichi nol
      pv->prev=*pend;//oldingi kontenerni yangi kontener bilan ulaymiz
```

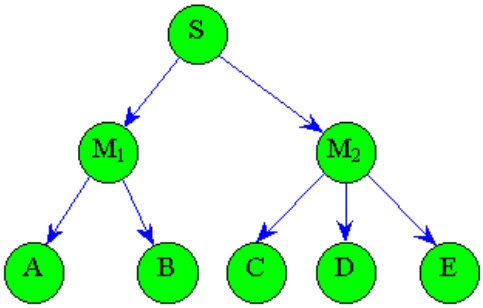
```
	(*pend)->next=pv;//oldingi kontenerni keyingi kontener bilan ulaymiz
	*pend=pv;//ro'yxat oxiriga yangi kontenerni qo'shamiz
}
Node *finder(Node *const pbeg,int key)
{
	Node *pv=pbeg;
	while(pv)
	{
		if(pv->data==key) break;//kontenerni ma'lumot yacheykasi berilgan
kalitga to'g'ri kelsa sikl to'xtaydi
		pv=pv->next;//kontenerni keyingi kontener bilan bog'laymiz
	}
	return pv;// topilgan kontenerni addresini qaytaramiz
}

bool removed(Node **pbeg,Node **pend,int key)
{
	if(Node *pkey=finder(*pbeg,key))//berilgan kalit bo'yicha qidirib
addressni olamiz
	{
		if(pkey==*pbeg)//berilgan address ro'yxat boshi bo'lsa
		{
			*pbeg=(*pbeg)->next;//ro'yxatni boshini ikkinchi element bilan
almashtiramiz
			(*pbeg)->prev=0;//ikkinchi elementdan oldingisini nolga
aylantiramiz
		}
		else if(pkey==*pend)//agar ro'yxat oxiri bo'lsa
		{
			*pend=(*pend)->prev;//oxirgi elementni bitta oldingi element
bilan almashtiramiz
			(*pbeg)->next=0;//oxirgi elementdan keyingi element mavjud
bo'lmaydi nolga aylantiramiz

		}
		else//ro'yxat o'rtasida bo'lsa
			{
			(pkey->prev)->next=pkey->next;
			(pkey->next)->prev=pkey->prev;
			}
		delete pkey;//dinamik ajratilgan joyni o'chirib
		return true;// amal bajarilgani uchun true qaytadi
```

```
        }
        return false;//aks holda false

}

Node *inserted(Node *const pbeg,Node **pend,int key,int d)
{
         if(Node *pkey=finder(pbeg,key))
        {
        Node *pv =new Node;
        pv->data=d;
        pv->next=pkey->next;
        pv->prev=pkey;
        pkey->next=pv;
        if(pkey!=*pend)(pv->next)->prev=pv;
        else *pend=pv;
        return pv;
        }
        return 0;
}
```



Binar daraxt ko’rinishidagi ma’lumotlar konteynerini yaratish.

Daraxt - bu chiziqsiz bog’langan ma'lumotlar tuzilmasidir.



3-rasm Binar daraxt

Daraxt o’zining quyidagi belgilari bilan tasniflanadi:

- daraxtda shunday bitta element borki, unga boshqa elementlardan murojaat yo’q. Mazkur elementga daraxt ildizi deyiladi;
- daraxtda ixtiyoriy elementga chekli sondagi ko’rsatkichlar yordamida murojaat qilish mumkin;
- daraxtning har bir elementi faqatgina o’zidan oldingi kelgan bitta element bilan bog’langan. Daraxtning har bir tuguni oraliq yoki terminal (barg) bo’lishi mumkin. Yuqoridagi chizmada M1, M2 - oraliq, A, B, C, D, E - barglardir. Terminal tugunning o’ziga xos tasnifi uning shoxlari yo’qligidir.

Balandlik - bu daraxt bosqichi soni. Yuqoridagi chizmadagi daraxt balandligi ikkiga teng.

Daraxt tugunlaridan chiqayotgan shohlar soni tugundan chiqish darajasi deyiladi (Keltirilgan chizmada M1 uchun chiqish darajasi 2, M2 uchun esa 3 ga teng). Daraxtlar chiqish darajasi bo’yicha sinflarga ajratiladi:

1) agar maksimal chiqish darajasi m bo’lsa, u holda bunday daraxt m-chi tartibli daraxt deyiladi;

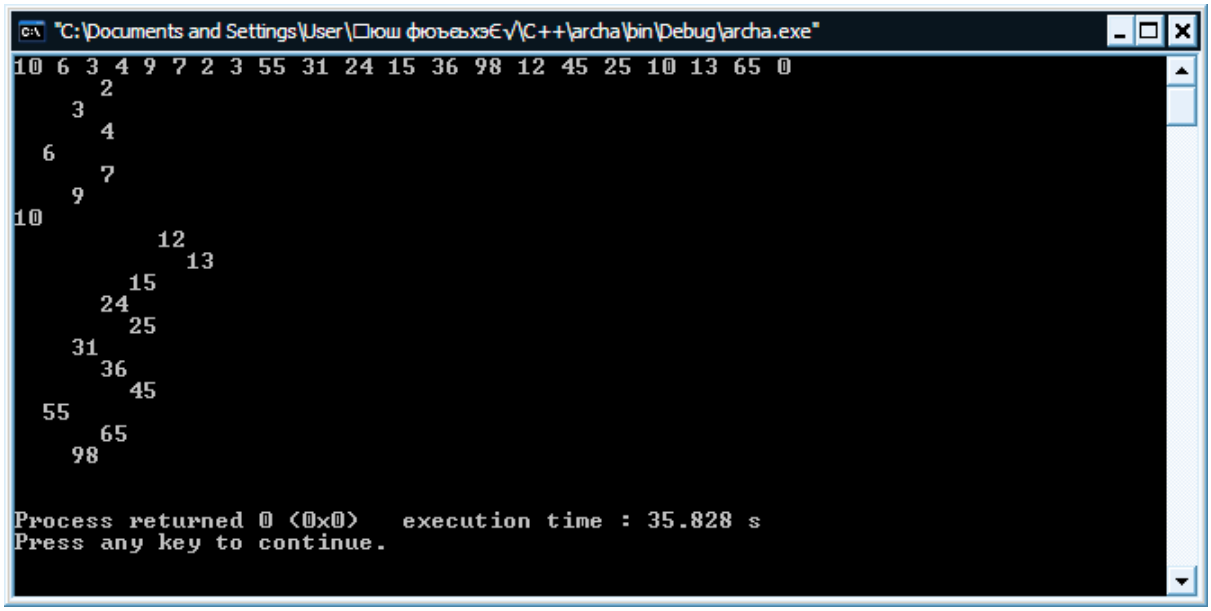
2) agar chiqish darajasi 0 yoki m bo’lsa, u holda to’liq m-chi tartibli daraxt bo’ladi;

3) agar maksimal chiqish darajasi 2 bo’lsa, u holda bunday daraxt binar daraxt deyiladi;

4) agar chiqish darajasi 0 yoki 2 bo’lsa, u holda to’liq binar daraxt deyiladi.

```
#include <iostream>

using namespace std;

struct Tree
{
int node;
Tree *left;
Tree *right;
};
Tree *first(int d)
{
	Tree *pv=new Tree;
	pv->node=d;
	pv->left=NULL;
	pv->right=NULL;
	return pv;
}
Tree *search_insert(Tree *root, int d)
{
	Tree *pv=root,*prev;
	bool found=false;
	while(pv&&!found)
	{
		prev=pv;
		if(d==pv->node) found=true;
		else if(d<pv->node) pv=pv->left;
		else pv=pv->right;
	}
	if(found) return pv;
	Tree *pnew=new Tree;
	pnew->node=d;
	pnew->left=pnew->right=NULL;
	if(d<prev->node)
		prev->left=pnew;
	else prev->right=pnew;
	return pnew;
}
void print_tree(Tree *p,int level)
{
	if(p)
```

```
        {
                print_tree(p->left,level+1);
                for(int i=0;i<level;i++)cout<<"    ";
                cout<<p->node<<endl;
                print_tree(p->right,level+1);
        }
}
void print(Tree *root)
{
        if(root)
        {
        cout<<root->node<<"    ";
        print(root->left);
        print(root->right);
        cout<<endl;
        }
}

int main()
{
        int a;
        cin>>a;
        Tree *root=first(a);
        cin>>a;
                while(a)
                {
                search_insert(root,a);
                cin>>a;
                }
          print_tree(root,0);
          cout<<endl;
          return 0;
}
```



Xulosa

Men ushbu kurs ishini bajarish davomida ma’lumotlar ustida ishlash. Kompyuterning RAM xotirasini boshqarish. Ularni modifikatsiya qilishni o’rgndim.

Kurs ishini bajarishda C++ dasturlash tili imkoniyatlari bilan kengroq tanishdim. Bundan tashqari STL yani standart shablonlar kutubhonasidagi ma’lumotlar toifalaridan foydalandim. Kurs ishini oxirida o’rganganlarim asosida mustaqil ma’lumotlar konteynerlarini yaratdim.

Foydalanilgan adabiyotlar.

1. Informatikadan maruzalar to’plami.
2. Шилдт Г. C++ Полное руководство. 2010.
3. http://google.com

4. ***Алфред В. Ахо., Джон Э. Хопкрофт, Джефри Д. Ульман.*** Структура данных и алгоритмы//Учеб.пос., М. : Изд.дом: "Вильямс", 2000, — 384 с.
5. ***Бакнелл Джулиан М.*** Фундаментальные алгоритмы и структуры данных в Delphi//СПб: ООО «ДиаСофтЮП», 2003. 560с.
6. ***Роберт Седжвик.*** Фундаментальные алгоритмы на C++. Анализ, Структуры данных, Сортировка, Поиск//К.: Изд. «ДиаСофт», 2001.-688 с.
7. ***Динман М.И.*** C++. Освой на примерах//СПБ.:БХВ-Петербург, 2006, 384.