#include <string.h>

// String - Xarakterler toplusudur

// String ozu arxada heapde yaradir

// char\* yaradir ve datalari silir

string text = "Hello world!";

int main()

{

string text = "";

text += "Apple";

text += "Apple";

text += "Apple";

cout << text << endl;

cout << text.size() << endl;

cout << text.length() << endl;

cout << text.capacity() << endl;

text += "Apple";

cout << text << endl;

cout << text.size() << endl;

cout << text.length() << endl;

cout << text.capacity() << endl;

cout << text.max\_size() << endl;

return 0;

}

OUTPUT:

AppleAppleApple

15

15

15

AppleAppleAppleApple

20

20

31

2147483647

// String her defe RAMA muraciet etmediyinden char massivinden (char\*) daha suretlidir

X) text.size() - xarakterlerin sayini qaytarir;uzunlugu qaytarir;

X) text.length() - uzunlugu qaytarir;xarakterlerin sayini qaytarir;

X) text.capacity() - minimal olaraq ilk tutumu 15 olur (string bos bele olarsa 15 yer ayiririr); 15i kecdikde tutum artir; bacardigi qeder elave yer ayirir; bir de yer gele biler,yeniden yer axtarmasin deye;

   capacity vectorda 2 qat artir; lakin burada her defe 16-16 artir; (ilk olaraq 15)

X) text.max\_size() - bir stringe maximum yazila bilecek xarakter sayidir (2147483647); daha cox lazimdirsa string massivinden istifade edirik;

int main()

{

string s(50, '\*'); // 50 deneli ulduz

cout << s << endl;

cout << s.capacity() << endl;

s.resize(10);

s.shrink\_to\_fit();

cout << s << endl;

cout << s.capacity() << endl;

return 0;

}

OUTPUT:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

63

\*\*\*\*\*\*\*\*\*\*

15

X) s.resize(x) - xarakterlerin sayini x - e azaldir;

X) s.shrink\_to\_fit() - umumi capcityden istifade olunmayan yerleri atir, temizleyir

int main()

{

string text = "Apple";

cout << text.capacity() << endl;

text.reserve(1000);

cout << text.capacity() << endl;

return 0;

}

OUTPUT:

15

1007

X) text.reserve(x) - x ve ya daha cox yer ayirir, reserve edir, capacity-ni deyisir

int main()

{

string text = "Happy programmers day";

cout << text << endl;

text.clear();

if (text.empty())

{

cout << "No Text" << endl;

}

else

{

cout << text << endl;

}

return 0;

}

OUTPUT:

Happy programmers day

No Text

X) text.clear() - stringi silir, temizleyir;

X) text.empty() - stringin bos olub olmadigini gosterir (bool qaytarir) (if - string a = ""; - true else - false)

int main()

{

string text = "Hi all";

cout << text << endl;

text += " , bye bye";

cout << text << endl;

text.append("Salam millet");

cout << text << endl;

return 0;

}

OUTPUT:

Hi all

Hi all , bye bye

Hi all , bye byeSalam millet

X) text.append() - text += Salam ile eynidir;

int main()

{

string name = "Elvin";

string surname = "Camalzade";

cout << name + " " + surname << endl;

return 0;

}

OUTPUT:

Elvin Camalzade

int main()

{

string text = "Happy";

cout << text[0] << endl;

cout << text.at(0) << endl;

text[0] = 'Z';

cout << text << endl;

for (int x = 0; x < text.size(); x++)

{

cout << text[x] << endl;

}

return 0;

}

OUTPUT:

H

H

Zappy

Z

a

p

p

y

X) String - xarakterler toplusu - arraydir - index anlayisi var;

X) text[x] ve text.at(x) - eynidir;

X) text[x] = 'Z'; x indexdeki elementi deyisir

int main()

{

string text = "Happy";

cout << text.front() << endl;

text.front() = 'P';

cout << text.front() << endl;

cout << text.back() << endl;

text.back() = 'i';

cout << text.back() << endl;

cout << text << endl;

return 0;

}

OUTPUT:

H

P

y

i

Pappi

X) text.front - ilk elementin referansini qaytarir

X) text.front() = 'P'; -  ilk elementi deyisdi

X) text.back() -  son elementin referansini qaytarir

X) text.back() = 'i'; -  son elementi deyisdi

int main()

{

string text = "Happy programmers day";

cout << text.substr(6,10) << endl;

cout << text << endl;

return 0;

}

OUTPUT:

programmer

Happy programmers day

X) text.substr(x,y); textin x indexinden y qabaga olan hisseni qaytarir

int main()

{

//insert ile - iteratorla(agilli class)isleyir - her nov container-le nece islemek lazim oldugunu bilir (ilk elementin adresi)

string text = "Happy programmers day";

cout << text << endl;

text.insert(text.begin(), 'E');

cout << text << endl;

text.insert(text.begin() + 5, 'E');

cout << text << endl;

text.insert(text.end() - 1, 'A');

cout << text << endl;

return 0;

}

OUTPUT:

Happy programmers day

EHappy programmers day

EHappEy programmers day

EHappEy programmers daAy

X) text.insert(text.begin() ve ya text.end() += 4, symbol or letter);

int main()

{

string text = "SALAM SALAM";

text.pop\_back();

cout << text << endl;

text.push\_back('Z');

cout << text << endl;

return 0;

}

OUTPUT:

SALAM SALA

SALAM SALAZ

X) text.pop\_back() - sonuncu elementi silir;

X) text.push\_back(const char x) - sona x - i artirir

int main()

{

string text1 = "salam";

string text2 = "Salam";

cout << boolalpha << (text1 == text2) << endl;

cout << boolalpha << text1.compare(text2) << endl;

return 0;

}

OUTPUT:

false

1

X) text1 == text2 - bir textin digerine beraber olmasini, eyni olmasi yoxlayir; Ideal yoxlama budur;

X) text1.compare(text2) - ascii xarakterlerinin cemine gore yoxlayir;0 - dirsa sozler beraberdir, eynidir

int main()

{

string text;

text = "Salam";

cout << text << endl;

string text2;

text2.assign("Salam");

cout << text2 << endl;

return 0;

}

OUTPUT:

Salam

Salam

X)  text = smth ve .assign() eynidir

int main()

{

string text = "Happy programmers day";

cout << text.find('j') << endl; // Olmayanda ekrana boyuk reqem qaytarir

cout << text.find('H') << endl; // Olanda ekrana ilk elementin indexini tapir

cout << text.rfind('j') << endl;

cout << text.rfind('y') << endl;

return 0;

}

OUTPUT:

X) .rfind() stringin sagindan isleyir

X) String input - cin >> text; stringin ilk bosluguna qeder - ilk qirilma noqtesine qeder - goturur

X) #include <string> - getline(cin, text); normal input

#include <initializer\_list>

//Test a(1,2,3,4,5,6); - classa bele gondermek ucun istifade olunur

#include <iostream>

#include <assert.h>

using namespace std;

template<class T,int size>

class FixedArray

{

T arr[size]{};

size\_t mysize = 0;

public:

FixedArray(initializer\_list<T> list)

{

for (auto data : list)

{

arr[mysize] = data;

mysize++;

}

}

void Show() const

{

for (int x = 0; x < mysize; x++)

{

cout << arr[x] << " ";

}

cout << endl;

}

T& operator[](int index)

{

return arr[index];

}

};

int main()

{

FixedArray<int, 20> myarray{1,3,4,6,2,6,2,6};

myarray.Show();

return 0;

}

X) initializer\_list - e indexle muraciet etmek olmur; for (auto data : list) islenir;

O) Dynamic data structure - datanin ramda ve ya databasede hansi formada saxlanmasi

   Data strukturlari

   1) Stack

   2) Queue

   3) Dequeue

   4) Priority Queue

   5) Circular

   6) Binary tree

   7) Linked list

   8) Double linked test

X) Stack - (heap, stack deyil) (FILO ile isleyir bu da)

Code Example :

// Custom stack class

#include <iostream>

#include <assert.h>

#include <initializer\_list>

using namespace std;

template<class T,int size>

class Stack

{

private:

T\* data;

size\_t size = 0;

public:

Stack() : data(NULL), size(NULL){}

void push(const T& value)

{

cout << " Value added successfully!" << endl;

T\* newarr = new T[size + 1]{};

for (size\_t x = 0; x < size; x++)

{

newarr[x] = data[x];

}

newarr[size] = value;

if (size != 0)

{

delete[]data;

}

data = newarr;

newarr = nullptr;

size++;

}

T pop()

{

assert(size > 0);

T\* newarr = new T[size - 1] {};

for (size\_t x = 0; x < size - 1; x++)

{

newarr[x] = data[x];

}

T last = data[size - 1];

if (size != 0)

{

delete[]data;

}

data = newarr;

newarr = nullptr;

size--;

return last;

}

// returns the last element

T& peek()

{

assert(size > 0);

return data[size - 1];

}

T& operator[](int index)

{

return data[index];

}

size\_t GetSize() const

{

return size;

}

void Clear()

{

if (this->data == NULL && this->size == NULL)

{

return;

}

delete[]data;

this->data = NULL;

this->size = NULL;

}

~Stack()

{

delete[]data;

}

};

class User

{

string name;

string surname;

public:

User()

{

}

User(const string& name, const string& surname)

{

this->name = name;

this->surname = surname;

}

void ShowUser() const

{

cout << "====== USER ====== " << endl;

cout << "Name ; " << this->name << endl;

cout << "Surname : " << this->surname << endl;

}

};

int\* main()

{

Stack<int, 10> mystack;

mystack.push(10);

mystack.push(11);

mystack.push(12);

//int lastint = mystack.pop();

//cout << " Last int : " << lastint << endl;

Stack<char, 10> mystack2;

mystack2.push('A');

mystack2.push('B');

mystack2.push('C');

char lastchar = mystack2.pop();

cout << " Last char : " << lastchar << endl;

Stack<User, 3> mystack3;

User u1("Tofiq", "Tofiqli");

User u2("A", "Ali");

User u3("John", "Johnlu");

mystack3.push(u1);

mystack3.push(u2);

mystack3.push(u3);

User lastuser = mystack3.pop();

cout << "Last User : " << endl;

lastuser.ShowUser();

return 0;

}

X) Queue (Normal sira ardicilligi ile isleyir mes. bank siralamasi h1, h2, h3), novbe anlayisi

Code example :

#include <iostream>

#include <assert.h>

#include <windows.h>

#include <initializer\_list>

using namespace std;

template<class T>

class Queue

{

T\* arr;

int capacity;

int front;

int rear;

int count;

public:

Queue(int size)

{

arr = new T[size] {};

capacity = size;

front = 0;

rear = -1;

count = 0;

}

void enqueue(T value)

{

assert(!IsFull() && "Queue is full");

arr[++rear] = value;

++count;

}

void dequeue()

{

assert(!IsEmpty() && "Queue is empty!");

auto temp = new T[count - 1]{};

for (int x = 0; x < count; x++)

{

temp[x] = arr[x + 1];

}

if (!IsEmpty())

{

delete[]arr;

}

arr = temp;

temp = nullptr;

--count;

}

bool IsFull()

{

return size() == capacity;

}

bool IsEmpty() const

{

return size() == 0;

}

T peek()

{

return arr[front];

}

int size() const

{

return count;

}

~Queue()

{

delete[]arr;

}

};

class Person

{

string name;

string surname;

double money;

public:

Person() = default;

Person(const string& name, const string& surname, const double& money)

{

SetName(name);

SetSurname(surname);

SetMoney(money);

}

string GetName() const

{

return name;

}

string GetSurname() const

{

return surname;

}

double GetMoney() const

{

return money;

}

void SetName(const string& name)

{

this->name = name;

}

void SetSurname(const string& surname)

{

this->surname = surname;

}

void SetMoney(const double& money)

{

this->money = money;

}

friend ostream& operator <<(ostream& out, const Person& p)

{

out << "======= PERSON INFO ======= " << endl;

out << "Name : " << p.GetName() << endl;

out << "Surname : " << p.GetSurname() << endl;

out << "Money : " << p.GetMoney() << endl;

return out;

}

};

 void main() {

Person p1("John", "Johnlu", 1000);

Person p2("Mike", "Eliyev", 200);

Person p3("Hakuna", "Matata", 100);

Queue<Person>persons(3);

persons.enqueue(p1);

persons.enqueue(p2);

persons.enqueue(p3);

while (!persons.IsEmpty())

{

auto person = persons.peek();

cout << person << endl;

Sleep(1500);

cout << "his work finished" << endl;

persons.dequeue();

}

/\*Queue myqueue(5);

int i = 0;

while (!myqueue.IsFull())

{

myqueue.enqueue(++i);

}

Sleep(2000);

while (!myqueue.IsEmpty())

{

cout << "Element : " << myqueue.peek() << endl;

myqueue.dequeue();

}\*/

}

X) Dequeue daginiq queue dir (ya baslangicdan cixir ya sondan cixir)

X) Priority Queue

O) Dynamic data structure - datanin ramda ve ya databasede hansi formada saxlanmasi

   Data strukturlari

   0) Array

   1) Stack

   2) Queue

   3) Dequeue

   4) Priority Queue

   5) Circular

   6) Binary tree

   7) (Singly-)Linked list

   8) Double linked test

X) Circular Queue (bezi standart datalar dovr edir)

X) Priority Queue (her hansi bir novbe var, neyin ustunluyu varsa o qabaga kecir)

X) (Singly-)Linked list (birlesmis list; bir element ozunde data ve ozunden sonraki elementin

  adressini saxlayir; bu formada elementeri biz muxtelif yerlerde saxlayiriq; next->next->next ile sondaki

  elemente cata bilerik; sondaki element ise next\* null,nullptr saxlayir; linked liste elementi oxumaq n defedir; insert() yeni pushfirst() -  emeliyyati linked listde daha suretlidir;

pushlast() bir bir nodlardan sonuncusunu axtarir deye yavasdir

  data oxuma(index ile) emeliyyatlari arrayda linked listden daha suretlidir;

  elave etmek ise linked listde daha suretlidir;

X) Double Linked list ( bir element ozunde data ve ozunden sonraki ve evvelki elementin adresini saxlayir)pushfirst() ve pushlast() - insert() suretlidir arraydan

X) Double Linked List

     Nod var (data ve adress saxlayir)

     Linked listde olan mentiqdir lakin bunda previous anlayisi da var (previous,next)

     Pushlast() - pushfirst() kimi emeliyyatlarin sayi 1 dir - O(1)

X) Binary tree

Elementler sort olunumus formada saxlanilir.

Ustunluyu diger data saxlanma strukturlarindan ferqi - datalar sira ile saxlanilir ve axtaris daha suretlidir (sirali datalar arasinda elementi tapmaq daha asandir)

En basda gelen element root elementdir.

Her bir elemtin 2 child elementi olur (2 elementin gosterir)

Mertebe anlayisi var binary tree de

left ve right nodu var

bir nodda sagdaki elementler ondan boyuk, soldaki elementler ondan kicik olmalidir

insert emeliyyati suretlidir

Big O Notation O(log(n))

tam agacda her nodun iki qolu olur

???

2^n+1 - 1

n - hundurluk

mertebe = hundurluk - 1

???

Object relationship

1) Composition (bir obyekt coxlu xirda hisselerden

   yaranirsa bu composition elaqesidir mes. masil

   masin yox olsa onun hisseleri de yox olmalidir

   masin hemin hisselerden asilidir) elaqe coxdur

2) Aggregation (obyekt digerinin icinde ola da biler olmayda da biler

   mes. her banka geden adamin bank hesabi olmasi mecbur deyil)

3) Association (bir obyekt digeri ile elaqe qurur lakin onun icinde olmasi mecbur deyil

   mes. patient doctor) elaqe daha azdir

4) Dependency (eger bir obyekt yarananda diger obyektin her hansi funksiyasini oyadirsa

   hemin obyekt digerinden asilidir(asililiq - dependency))

Nested (Embedded) Classes

{

class List

{

public:

private:

class Node

{

public:

int data;

Node\* next;

Node\* prev;

};

private:

Node\* head;

Node\* tail;

};

Node tipi yalniz List classinin daxilinde gorunur

Oxuynarligi artirmaq ucundur

Ex.

class Product

{

string name;

public:

struct ProductType

{

enum Types

{

FANCY,AWESOME,USEFUL

};

};

struct ProductBoxType

{

enum Types {

BOX,BAG,CRATE

};

};

Product(ProductType::Types type, ProductBoxType::Types boxtype,string name)

{

}

};

void main()

{

Product p(Product::ProductType::AWESOME, Product::ProductBoxType::BOX, "APPLE");

}

}

Inheritance (bir classin digerinden torenmesi)

{

struktura qazandirir;

yigcamliq qazandirir;

kod artiqliginin qarsisini alir;

Every derived class is base class;

But not every base class derived class;

base class/ super class/ parent class;

derived class/ subclass/ child class/

Istenilen classdan istenilen classi kodda toretmek olar, lakin mentiqi uygunluq olmalidir;

}

Inheritance toreme demekdir ve novleri var

1) Simple Inheritance

2) Multilevel Inheritance

3) Hierarchical (arranged in order of rank) Inheritance

4) Multiple Inheritance

1) Simple Inheritance

Sade toremedir mes. animal classi var, odna catt, dog ve s. torenir

        Animal

       /           \

     Cat         Dog

2) Multilevel Inheritance

        Animal

       /           \

     Cat         Dog

      /

   Tiger

Tiger is a Car, and an animal. (Every derived class is base class)

Bu Multilevel Inheritance.

Tiger hem oz, hem Cat-in,hem de animal-in ozelliklerini dasiyir

3) Hierarchical (arranged in order of rank) Inheritance

Hierarchical Inheritance binary treeye benzeyir

Adından da göründüyü kimi, bu, siniflərin iyerarxiyasıdır(siralanmasi). Tək əsas sinif və çoxlu törəmə siniflər var. Bundan əlavə, törəmə siniflər bəzi digər siniflər də tərəfindən törənir Beləliklə, ağaca bənzər bir quruluş iyerarxiya şəklində formalaşır.

               Animal

       /                          \

     Cat                       Dog

    /     \                     /      \

Tiger  Scot         Pitbull    Golden

4) Multiple Inheritance

Bir klass 2 ve daha artiq klassdan torenir

 Multiple Inheritance, bir sinif birdən çox əsas sinifdən törəndikdə baş verir. Beləliklə, sinif çoxlu irsiyyətdən istifadə edərək bir neçə əsas sinifdən xüsusiyyətləri miras ala bilərş

        Human

      /              \

  Teacher       Student

     \                  /

     Superstudent

Bezi konfliktler bas vere biler

1) mes. 2 base classda eyni adli fieldler var ve onlar derived classda toqqusa biler bunu qarsisini almaq ucun SCOPE RESOLUTIONDAN istifade edirik (BASE1::field, BASE2::field)

2) Diamond problem (solution - "virtual")

Problem:

A    A

 |      |

 B    C

  \  /

   D

Burada D ni yaradarken 2 defe A constructoru cagirilir

onun qarsisini almaq ucun B ve C classlarinin qarsisina virtual acar sozu yazilir

Solution with the keyword VIRTUAL

   A

 /   \

B     C

 \   /

   D

class A

{

public:

A()

{

cout << " Default constructor of A " << endl;

}

};

class B: virtual public A

{

};

class C : virtual public A

{

};

class D : public B, public C

{

};

int main()

{

D d;

return 0;

}

Neticede bir defe A constructoru cagirildi

OUTPUT:

 Default constructor of A

Notes:

1)Derived classda bos obyekt yaratmaq ucun base classda bos obyekt yaranmalidir

2)Private fieldler toreme klassda hemise inaccessible (private) olur

3)Torenen klass override edende (base classda olan metodu toreme classda eyni adla yazmaq) artiq base classdaki hemin metodun evezine toreme classdaki hemin override edilmis metod cagirilir. Override-in olmasi ucun virtual acar sozu ve base classa pointer olmalıdır

Polimorfizm - çoxşəkillilik deməkdir

Bir obyektin funksiyasinin ozunu basqa obyektlerde ferqli aparmasi polimorfizmdir.

Mes. boyukle basqa cur danisirsan, 5 yasli usaqla basqa

Polimorfizmin olmasi runtime zamani bas verir

Inheritance vasitesiyle base classin pointerinde o ve onun toremelerinin tiplerini saxlaya biler;

Yeni :

Animal\* animal = new Car(); yaza bilerik

(ozunden gelen,torenen tip)

Inheritancein verdiyi mohtesemlik:

Inheritance vasitesiyle bir massivin icinde muxtelif tipli, lakin eyni basse classdan torenmis elementler saxlaya bilerik

///////////////////////////////////////////////////////////////////////////////////////////////

virtual acar sozu ile override edirik (yeni metodu basqa obyekte gore muxtelif cur reallasdirdi);

Meselen;

class Base

virtual string GetName()

{

           return "I am base";

}

class Derived :public Base

string GetName()

{

         return "I am derived";

}

main()

       Base b(10);

       Base\* base = new Derived(20);

       cout << base->GetName() << endl;

OUTPUT:

I am derived

///////////////////////////////////////////////////////////////////////////////////////////////

//\*Metodlar tamamile eyni olmalidir((const)//

///////////////////////////////////////////////////////////////////////////////////////////////

Klasslar arasi dinamik polimorfizmin olmasi ucun base classa pointer saxlamaliyiq

Hansi metodun muxtelif olmagini isteyirikse qarsisina virtual yaziriq

Inheritancein bize verdiyi komekliklerden biri:

base pointer saxlamaqla basein torenenlerini massivde saxlaya bilerik

///////////////////////////////////////////////////////////////////////////////////////////////

1)Early binding or Static Polymorphism :

Statik polimorfizmdir, kompliyasiya vaxtinda neyi cagiracagini bilir

Mes.

void Show()

{

cout << 10 << endl;

}

main()

{

Show();

}

2) Late binding or Dynamic Polymorphism :

 program run olandan sonra, run timedan sonra neyi cagiracagini bilir

Mes.

void add(int a, int b) {

cout << a + b << endl;

}

main()

{

void(\*ptr)(int, int) = add;

ptr(10,20);

}

///////////////////////////////////////////////////////////////////////////////////////////////

Polimorfizmin olmasi ucun inheritance lazimdir mi?

Yox, cunki funksiyaya pointer saxlamaqla late binding ede bilerik

///////////////////////////////////////////////////////////////////////////////////////////////

Torenen klasslara overrride ederken, hemin overridein tamamile dogru oldugun qaranti etmek ucun hemin methodun

qarsisindan override acar sozunu yaza bilerik (yazmasaqda override edir lakin qaranti etmek ucun, hemcinin oxunarligini artirmaq ucun yazila biler)

///////////////////////////////////////////////////////////////////////////////////////////////

A (virtual)

B:A

C:B

virtualliq(override) C-ye qeder (sona kimi) torenir ve A dan baslayir

///////////////////////////////////////////////////////////////////////////////////////////////

A (virtual)

B:A (final)    (sintaksis - void Speak() override final)

C:B

final keyword oldugu ucun virtualliq(override) B ye qeder torenir

///////////////////////////////////////////////////////////////////////////////////////////////

Abstraction

////////////////////////////////////////////////////////////

(Classda konkretlik yoxdursa o yalniz base class ola biler ve ona pointer saxlaya bilerik.)

////////////////////////////////////////////////////////////

Base class rolunu oynasin deye abstract classlar yaradiriq ve onlarin obyekti yaranmir, konstruktorunu yaza bilerik lakin obyekti yaradilmir-ona pointer saxlaya bilerik

////////////////////////////////////////////////////////////

klassin abstract olmasi ucun onun icinde en azi 1

 pure virtual method olmalidir

 pure virtual method yoxdursa classi abstract etmek ucun

pure virtual destructor ede bilerik

Mes.

class Figure

{

protected:

double area;

public:

virtual void WhatIsMyShape() = 0;

};

////////////////////////////////////////////////////////////

klassin icinde olan metodlar her sey pure virtualdirsa o INTERFACEdir

////////////////////////////////////////////////////////////

pure virtual destructor olduqda derived classda destructor ozu override olur

////////////////////////////////////////////////////////////

Exceptionlari istenilen vaxt ata bilerik

Proqramin cokmesinin qarsisini aliriq

Meqsedle exception atmaq neye lazimdir?

Bir seyi funksiya tapmadiqda --1 qaytarirdiq

Lakin bu detalli deyildi

Exception attiqda daha detalli melumat qaytara, zamanini ve cixdigi yeri, fayli, kodun setrini bile bilerik

class string int enum ve s. throw ede bilerik(exception ara bilerik)

exception funksiyanin icinden atila biler ve handle ede bilerik (try catch ile)

try

{

       dosmth(-10);

}

catch (const char\* ex)

{

       cout << "Error : " << ex << endl;

}

catch (int ex)

{

       cout << "Error : " << ex << endl;

}

bu const char\* ve int i handle ede biler

Exception atilandan sonraki kod (catcha kimi) islemir

catchi tapdiqda ora girir tapmasa error

Eger basqa tipli elementlere esases exception atiriqsa bele ede bilerik (uc noqte) - (...)

void main()

{

try

{

dosmth(2);

}

catch (int)

{

cout << "Int Error" << endl;

}

catch (const char\*)

{

cout << "Const char error" << endl;

}

catch (...)

{

cout << "General Error" << endl;

}

}

Pis terefi errorun ne oldugunu bilmirik

throw-dan sonraki hec bir kod bloku islemir

Stack Unwiding (Exception)

exception atilan yerden(throwdan) tutulan yere(catcha) kecdikde steakde throwdan

evvel yaranan elementleri (Stack Unwiding) ozu silir

#include <iostream>

using namespace std;

class Exception

{

protected:

string text;

string source; // file

int line;

string date;

public:

Exception(string text,string source, int line, string date)

:text(text),source(source),line(line),date(date)

{

}

virtual void PrintMessage() const

{

cout << " ========= ERROR INFO ========= " << endl;

cout << " Error Content : " << text << endl;

cout << " Source : " << source << endl;

cout << " Line Number : " << line << endl;

cout << " Date : " << date << endl;

}

};

class OutOfRangeException :public Exception

{

public:

OutOfRangeException(string text, string source, int line, string date)

:Exception(text, source, line, date)

{

}

void PrintMessage() const override

{

cout << " Error type : " << "OutOfRangeException" << endl;

Exception::PrintMessage();

}

};

class InvalidArgumentException : public Exception

{

public:

InvalidArgumentException(string text, string source, int line, string date)

:Exception(text, source, line, date)

{

}

void PrintMessage() const override

{

cout << " Error type : " << "InvalidArgumentException" << endl;

Exception::PrintMessage();

}

};

template<typename T>

class Array

{

private:

T\* data;

int size;

public:

int GetSize() const

{

return size;

}

Array<T>& operator=(const Array<T>& other) = delete;

Array(const Array<T>& other) = delete; // kimsenin copy assignment (basqa seyler de ola biler) etmesine icaze vermir

Array(int size) throw(InvalidArgumentException) // Bu kod blokunda cixa bilecek exceptionlari gosterir

{

if (size < 0 || size > 1000000)

{

throw new InvalidArgumentException("Size should be between 0 and 1000000",

\_\_FILE\_\_, \_\_LINE\_\_, \_\_DATE\_\_);

}

this->size = size;

data = new T[size]{};

}

T& operator[](int index) throw(OutOfRangeException) // Bu kod blokunda cixa bilecek exceptionlari gosterir

{

if (index < 0 || index >= size)

{

throw new OutOfRangeException("Index should be more than 0 and less than " << size,

\_\_FILE\_\_, \_\_LINE\_\_, \_\_DATE\_\_);

}

return data[index];

}

};

int main()

{

try

{

Array<int> a(-10);

}

catch (const Exception \*ex)

{

ex->PrintMessage();

}

return 0;

}

#include <typeinfo> - datanin tipini tapmaq ucun library

int i = 5;

float j = 1.5f;

char c = 'a';

const type\_info& t1 = typeid(i \* j);

const type\_info& t2 = typeid(i \* c);

const type\_info& t3 = typeid(c);

cout << "t1 is of type " << t1.name() << endl;

cout << "t2 is of type " << t2.name() << endl;

cout << "t3 is of type " << t3.name() << endl;

//////////////////////////////////////////////////////////////////

Bir klassin polimorfik klass olmasi ucun en azi 1 virtual acar sozu olmalidir

//////////////////////////////////////////////////////////////////

mutable int roll2; // const daxilinde deyismek ucun mutable istifade olunu

//////////////////////////////////////////////////////////////////

C++ supports 4 types of casting

1) static cast

2) dynamic cast

3) Const cast

4) Reinterpret cast

1) static cast - (static\_cast)

Kompliyasiya vaxtinda cevirmeni yoxlayir

Cevire bilirse error cixartmit

Cevire bilmirse error cixartir (compile error)(red line)

castinge qarantiya verir

2) dynamic cast - (dynamic\_cast)

Run time merhelesinde cevrilib cevrilmeyeceyini bilirik

cevire bilmedikde nullptr qaytarir

toreme classin ozelliklerine catmaq ucun

dynamic\_cast istifade oluna biler

upcast(derived -> base)

downcast(base -> derived)

3) Const cast

Const cast constant olan pointeri constantliqdan cixarmaq ucundur

Example :

class Student

{

int roll;

mutable int roll2; // const daxilinde deyismek ucun mutable istifade olunu

public:

Student(int r) : roll(r)

{

}

void Fun() const

{

//this->roll = 20 // error

this->roll2 = 200; // okay because roll2 is mutable

(const\_cast<Student\*>(this))->roll = 20;

}

};

4) Reinterpret cast

Elinden gelen her seyi edir ki cevirsin

Neyise cevire bilirse cevirir, cevire bilmirse qeribe data qaytarir;

yaxsi ceheti : cevire bilmese bele datani qoruyub saxlayir

string - containerdir, datani burada saxlayiriq

stream string - axindir, datani burada saxlamaq olmur

///////////////////////////////////////////////////////////

butun streamlerin koku ios\_base den gelir

///////////////////////////////////////////////////////////

Stream - axindir

///////////////////////////////////////////////////////////

Consola streamen novleri

cout(console out)

cin(console in)

cout >> "Hello" << endl;

cout <-- Hello

Hello console out a gedir

///////////////////////////////////////////////////////////

cin << number

cin --> number

cin numbera gedir

///////////////////////////////////////////////////////////

int num = -10;

cout << showpos << num << endl;

OUTPUT:

-10

///////////////////////////////////////////////////////////

int num = 10;

cout << showpos << num << endl;

OUTPUT:

+10

///////////////////////////////////////////////////////////

bool result = true;

cout << boolalpha << result << endl;

OUTPUT:

1

///////////////////////////////////////////////////////////

bool result = true;

cout << boolalpha << result << endl;

OUTPUT:

true

///////////////////////////////////////////////////////////

cout << "|" << setw(5) << left << 1 << "|" << endl;

cout << "|" << setw(5) << right << 1 << "|" << endl;

birincide data soldan, ikincinde sagdan gosterilir

OUTPUT:

|1    |

|    1|

///////////////////////////////////////////////////////////

#include <sstream> // string stream

string stream - de data saxlanilmir

Bir defe oxuduq, ondan sonra onun gore bilmeyeceyik

daha cox yaddasa qenaet etmis oluruq string streamle

///////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////

#include <iostream>

#include <sstream>

using namespace std;

void main()

{

stringstream os;

os << "Hello World and Programmers";

string data = "";

os >> data;

cout << data;

os >> data;

cout << data;

os >> data;

cout << data;

os >> data;

cout << data;

}

///////////////////////////////////////////////////////////

void main()

{

stringstream os;

os << "Hello World and Programmers" << endl;

string data;

getline(os, data);

cout << data << endl;

}

///////////////////////////////////////////////////////////

void main()

{

stringstream os;

os << "Hello World and Programmers \n Today is an unhappy day" << endl;

string data;

getline(os, data);

cout << data << endl;

getline(os, data);

cout << data << endl;

}

C++ files

File ile islemek ucun

#include <fstream>

Text File Modes:

1) out write (filename,ios::out) (while writing filetype should be ofstream)

2) in read (filename,ios::in) (while reading filetype should be ifstream)

3) app (append) (filename,ios::app) (while appending filetype should be ofstream)

Binary File Modes:

1) out write (filename,ios\_base::binary | ios\_base::app) (while writing filetype should be ofstream)

2) in read (filename,ios\_base::binary)

3) app (filename,ios\_base::binary)

SMART POINTERLERIN 3 NOVU VAR

1. Shared Pointers
2. Unique Pointers
3. Weak Pointers

WEAK POINTERS

Onlar cox istifade olunmur, cunki general problemleri hell ede bilmir, adi pointerler kimidirler

Unique Pointers

Bir adressi bir pointer gostere biler. Onun ucun pointerlerin copy assignment ve copy constructoru bloklayiriq.

COPY ASSIGNMENT, COPY CONSTRUCTOR

int\* a = new int(100);

int\* b = a;

int\* c = b;

Shared Pointers

Her hansi bir adressi birden cox pointer gosterir. O, eger her hansi pointerin isi bitibse ve basqa pointerler de o datani gosterirse onu silmir. Artiq pointer gostermedikde ve data ile sonuncu pointerin isi bitdikde datani silir. (Pointerlirin sayini counterde saxlayacagiq)

haziri:

shared\_ptr<int>a(new int(42));

shared\_ptr<int>b(a);

STL - standart template library.

STL - de Containerlerin var ve bunlar templatedir

Istenilen containerin begin ve end i var (Containerlerin ortaq ceheti)

Icinde bezi toollarda var, hansilar ki, alqoritmlerle

ve containerlerle isleye bilir

1) Vector

Containerdir, array mentiqi ile isleyir (mes. indexler muraciet etmek olur)

Arxada oz arrayini yaradir, dinamik yaddas sahesi ile isleyir

Eger arraya 4 element elave etmisinizse, o basqa yerler de ayirir

Elave yerler ayirir ki, basqa data gelende orada saxlasin. Her defe

yeni array yaratmaqdansa (bunu etmek yaxsi deyil(yaddasda yer axtarmaq))

bunu edir

Vector istifade etmek ucun #include <vector>

Iterator agilli klassdir ve her bir containerde nece gezmek lazim oldugunu bilir

Kompüter proqramlaşdırmasında iterator proqramçıya

konteyneri, xüsusən də siyahıları keçməyə imkan verən

obyektdir. Müxtəlif növ iteratorlar tez-tez konteynerin

interfeysi vasitəsilə təmin edilir.

2) List

STL Basqa containerler (Lesson 24 - vector, list kecmisdik)

1) set

2) map

3) multiset

4) multimap

To use:

#include <set>

#include <map>

1) Set

Set in arxasinda binary tree durur

ve setde elementler tekrarlanmir, her elementi unikal

olaraq saxlamaga calisir

Set elementleri sort edib saxlamaga calisir

2) Map

Map in arxasinda binary tree durur

En cox islenen map dir - key(.first) value(.second) mentiqi ile isleyir

Ve key lere gore sort edib saxlayir

Keyler unikal olur her dilde

ypx["10-KY-248"] = "Range Rover";

"10-KY-248" mapda varsa valuesini deyisir, yoxdursa value kimi "Range Rover"-i

key kimi "10-KY-248"-i elave edir

structda map saxlamaqla onun ozellik sayini deyisken saxlaya bilerik

3) Multiset

Setle eynidir lakin unikalliq yoxdur

4) Multimap

melumat saxlamaq ucundur

indeksle catmaq olmur  : //cout << multimap["AUDI"] << endl; // error

unikalliq yoxdur

////////////////////////////////////////////////////////////////////////

#include "list" (next, previous (double linked list))

#include <forward\_list> (next (single list))

////////////////////////////////////////////////////////////////////////

Predicates - qisa bool qaytaran funksiya

listin sort metodunda istifade oluna biler prediktalar

sorta predikatin (funksiyanin) adresini gonderirik

// Predicates - qisa bool qaytaran funksiya

bool byNameAsc(const Kitty& k1, const Kitty& k2) // - Predicate

{

return k1.GetName() < k2.GetName();

}

bool byAgeAsc(const Kitty& k1, const Kitty& k2) // - Predicate

{

return k1.GetAge() < k2.GetAge();

}

void main()

{

list<Kitty> kitties;

kitties.push\_back(Kitty("Toplan", "Toplan123", 35));

kitties.push\_back(Kitty("Mestan", "Qara", 12));

kitties.push\_back(Kitty("Black", "Jav Jav", 67));

kitties.push\_back(Kitty("Garfield", "Hungry Cat", 16));

PrintKitties(kitties);

kitties.sort(byNameAsc);

cout << endl;

PrintKitties(kitties);

kitties.sort(byAgeAsc);

cout << endl;

PrintKitties(kitties);

}

functor - ozunde predikat saxlayan, daxilinde bool qaytaran () overloading olan klassdir

functorlar yigcamdir

functor yazmaq ucun moterizeni () overloading edirik

mes.

class byAge

{

bool order;

public:

byAge(bool order) : order(order)

{

}

bool operator()(const Kitty& cat1, const Kitty& cat2)

{

if (order)

{

return cat1.GetAge() < cat2.GetAge();

}

return cat1.GetAge() > cat2.GetAge();

}

};

void main()

{

     kitties.sort(byAge(false));

}

/////////////////////////////////////////////////////////////////////////////////////////

functor bir nece predikat yazmaqdansa istifade oluna biler

/////////////////////////////////////////////////////////////////////////////////////////

basqa data tipinde de data qaytaran predikatlar olur (mes int), bool qaytaran qisa funskiya yalniz onun butun dillerdeki izahidir

/////////////////////////////////////////////////////////////////////////////////////////

transfrom() - containerin uzerinde gezir ve sizin verdiyiniz functor ve ya predikata esasen emeliyyat aparor

/////////////////////////////////////////////////////////////////////////////////////////

#include <numeric>

accumulate() sonda emeliyyat gonderirik onu edir (default toplamaqdir)

count() - datanin vectordaki sayini qaytarir

find() - datani tapsa adresini qaytarir

binary\_search() -  tapsa true, tapmasa false qaytarir (vectorun sortlanmasi lazimdir)

/////////////////////////////////////////////////////////////////////////////////////////

for\_each(v.begin(), v.end(), print); - meqsedi vectorun uzerinde gezmekdir ve ne etmek istediyimiz ozumuz gonderirik\

/////////////////////////////////////////////////////////////////////////////////////////

// containerde 10dan kicik olan elementi qaytarir

auto p = find\_if(v.begin(), v.end(), [](int data)->bool {

return data < 10;

});

/////////////////////////////////////////////////////////////////////////////////////////

sort(v.begin(), v.end(), [](int a, int b)->bool {

return a > b;

});

/////////////////////////////////////////////////////////////////////////////////////////

int c = count\_if(s.begin(), s.end(), [](Student s) {

return s.GetScore() > 90;

});

namespace blokdur

1) kodlari qruplasdirmaq ucundur

2) iki eyni adli seylerin toqqusmamasi ucundur

mes. windows.h daki rectangle ile oz rectangle classimiz toqqusa biler

3) kodun oxunarligini artirmaq ucundur

header file namespacesiz yazilan kodlar global namespace dedir

global namespacede her sey bir birini gorur

namespace std

{

}

eger std varsa hemin namespace skoplarin icinde yazdiqlarimizi elave edir

yoxdursa yenisini yaradir

namespace Network

{

namespace HTTP

{

namespace HttpClient

{

class HttpClientRequest

{

};

}

}

}

// Qisaltma

namespace NHH = Network::HTTP::HttpClient;

void main()

{

//Network::HTTP::HttpClient::HttpClientRequest client;

        Yuxardakinin evezine asagidakini yazmaq olar

NHH::HttpClientRequest client2;

}