**Лабораторна робота № 1**

**Тема:**Анатомія класів. Робота з класами

**Мета:** Закріпити знання про базові поняття класу, поняття про конструктори і деструктори.

**Завдання 1.** Створіть клас Int, який імітуватиме стандартний тип int. Єдине поле цього класу повинно мати значення типу int. Створіть методи, які будуть встановлювати значення поля рівним нулю, ініціалізувати його цілим значенням, виводити значення поля на екран і складати два значення типу Int.

Напишіть програму, в якій будуть створені три об'єкти класу Int, два з яких будуть ініціалізованими. Додайте два ініціалізованних об'єкта, надайте результат третьому, а потім відобразіть результат на екрані.

#include <iostream>

using namespace std;

class **Int**{

public :

int number;

**Int**(int n){

number = n;

}

void **put**(int n = 0){

number = n;

}

void **Print**(){

cout << number;

}

int **sum**(Int n){

number += n.number;

return number;

}

};

int **main**()

{

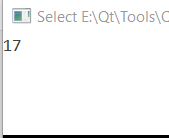
Int a = 9, b = 8;

Int c = a.sum(b);

c.Print();

return 0;

}



**Завдання 2.**

#include <iostream>

#include <conio.h>

using namespace std;

class **Kasa**{

public :

unsigned int amountCars;

double Payment;

**Kasa**(){

amountCars = 0;

Payment = 0;

}

void **display**(){

cout << amountCars << " " << Payment << endl;

}

void **payingCar**(){

amountCars++;

Payment += 0.5;

}

void **nopayingCar**(){

amountCars++;

}

};

template <typename A>

class **Test**

{

public:

A testA;

};

int **main**()

{

Test<Kasa> a;

a.testA = \*(new Kasa());

Kasa kasa;

while (true) {

switch (getch()) {

case 27:

kasa.display();

return 0;

case 112:

kasa.payingCar();

break;

case 110:

kasa.nopayingCar();

}

}

return 0;

}

**Завдання 3.**

#include <iostream>

using namespace std;

class **time**{

public :

int hour;

int minute;

int second = 0;

**time**(){

hour = 0;

minute = 0;

second = 0;

}

**time**(int \_hour, int \_minute, int \_second):time(){

second += \_second;

if(second >= 60){

minute += second/60;

second = second%60;

};

minute += \_minute;

if(minute >= 60){

hour +=minute/60;

minute = minute%60;

};

hour += \_hour ;

}

void **Display**(){

cout << hour << ":" << minute << ":" << second << endl;

}

void **Sum**(time t){

second += t.second;

if(second >= 60){second -=60; minute++;};

minute += t.minute;

if(minute >= 60){minute -=60; hour++;};

hour += t.hour;

}

};

int **main**()

{

time t1(3, 40, 60), t2(2, 20, 5);

t1.Display();

t2.Display();

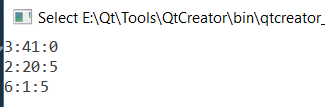
time t3;

t3.Sum(t1);

t3.Sum(t2);

t3.Display();

return 0;

}

**Висновок**: закріпив знання про базові поняття класу, поняття про конструктори і деструктори.

**Лабораторна робота № 2**

**Тема:**Робота з лінійними списками. Конструктор і деструктор класу

**Мета:** Навчитись використовувати конструктори і деструктори класів,  створювати класи для опису лінійних списків

**Завдання 1.**Реалізувати один простий клас згідно варіанту індивідуального завдання, що містить закриті данні, а саме два типа даних: числове та рядкове, реалізоване через покажчик на char. Потрібно створити декілька екземплярів класу статично і динамічно, а також масив та продемонструвати дію всіх конструкторів і методів.

#include <iostream>

#include <string.h>

using namespace std;

//Lab2 PR

class **Toys** {

char \*Owner;

int Old;

public:

**Toys**();

**Toys**(char \*, int);

**Toys**(const Toys&);

void **SetOwner**(char\*);

char\* **GetOwner**();

int **Getchar**();

void **Print**();

void **Input**();

~**Toys**();

};

Toys::**Toys**() {

Toys::Owner = nullptr;

Toys::Old = 0;

}

Toys::**Toys**(char \*\_Owner, int Old) {

Toys::Owner = \_Owner;

Toys::Old = Old;

}

Toys::**Toys**(const Toys& T) {

Toys::Owner = new char[40];

for (int i = 0; i < 40; i++){

Owner[i] = T.Owner[i];

}

Toys::Old = T.Old;

}

void Toys::**SetOwner**(char\* Owner) {

Toys::Owner = Owner;

}

char\* Toys::**GetOwner**() {

return Toys::Owner;

}

int Toys::**Getchar**() {

return \*Toys::Owner;

}

void Toys::**Print**() {

cout << Toys::Owner << endl << Toys::Old << endl;

}

void Toys::**Input**() {

char a[40];

cin >> a;

Owner = new char[40];

for (int i = 0; i < 40; i++){

Owner[i] = a[i];

}

cin >> Toys::Old;

}

Toys::~**Toys**() {

delete Owner;

}

int **main**()

{

char str [] = {"djagblkablk"};

Toys A, \*B = new Toys(str, 6);

A.Input();

A.Print();

B->Print();

Toys C = A;

Toys mas[] = { A, \*B, C};

for (int i = 0; i < 3; i++) {

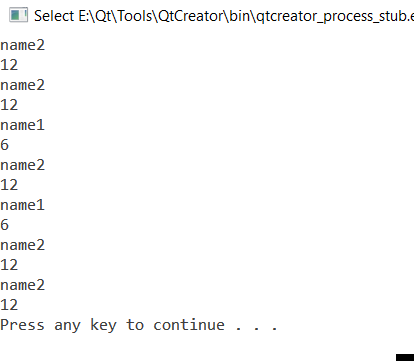
mas[i].Print();

}

C.Print();

system("pause");

return 0;

}

**Завдання 2.**Створити клас для реалізації однозв’язного списку з динамічним виділенням пам’яті типу “черга” та методи додавання числа в чергу та вилучення з неї.

#include <QCoreApplication>

#include <iostream>

using namespace std;

class **Queue**{

public:

**Queue**(int n):number(n), next(nullptr){};

int number;

Queue \*next;

void **Print**(int n){

if(n == 0)cout << number << endl;

else next->Print(n-1);

}

void **add**(int n){

if (next != nullptr) next->add(n);

else next = new Queue(n);

}

void **remove**(int n){

if (n = 0){

number = next->number;

remove(1);

}

if(n == 1) next = next->next;

else next->remove(n-1);

}

};

int **main**(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

Queue e(0);

e.add(1);

e.add(2);

e.add(3);

e.add(4);

e.add(5);

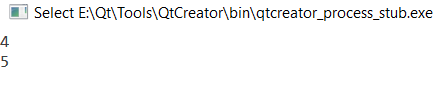
e.Print(4);

e.Print(5);

e.remove(4);

e.Print(4);

return a.exec();

}

**Висновок**: Навчився використовувати конструктори і деструктори класів,  створювати класи для опису лінійних списків

***ЛАБОРАТОРНА РОБОТА №3***

**Тема:** Перевантаження операцій класу

**Мета:** ознайомитись зі способами перевантаження операцій та навчитись використовувати їх при роботі з об’єктами.

**Завдання 3.**

#include <QCoreApplication>

#include <iostream>

using namespace std;

class **Int**{

int number;

public :

**Int**(long double n){

if(n > 2147483684){

cout << "error" << endl;

}

number = (int)n;

}

void **Print**(){

cout << number << endl;

}

void **put**(int n = 0){

number = n;

}

Int operator+(Int &a){

return Int(number + a.number);

}

Int operator-(Int&a){

return Int(number-a.number);

}

Int operator\*(Int &a){

return(number\*a.number);

}

Int operator/(Int &a){

return Int(number/a.number);

}

};

int **main**(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

Int ar(30000000000000);

Int b(3);

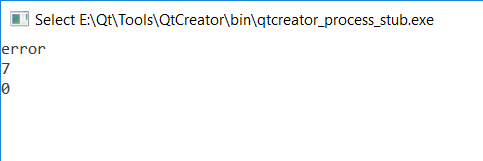
Int c(4);

Int d = b+c;

d.Print();

ar.Print();

return a.exec();

}

**Завдання 4.** Створити клас – вектор, який має у закритій частині вказівник на дані цілого типу та кількість елементів. Визначити необхідні конструктори, деструктор. Перевантажити потокові операції введення і виведення елементів вектора, операції + , – , \* , = та [ ] .

#include <QCoreApplication>

#include <iostream>

#include <string.h>

using namespace std;

class **Vector**{

int \*a;

int k;

public:

**Vector**(){

k = 0;

a = NULL;

}

**Vector**(int mas[], int k){

this->k = k;

a = new int[k];

for(int i = 0;i< k ;i++) a[i] = mas[i];

}

Vector operator+(int p){

k +=1;

int \*a = new int[k];

for (int i = 0 ; i < k-1;i++){

a[i] = this->a[i];

}

a[k-1] = p;

this-> a = a;

return \*this;

}

Vector operator-(int p){

for(int i = 0; i < k ;i++) a[i] -= p;

return \*this;

}

Vector operator\*(int p){

for(int i = 0; i < k ;i++) a[i] \*= p;

return \*this;

}

Vector operator/(int p){

for(int i = 0; i < k ;i++) a[i] /= p;

return \*this;

}

int\* operator=(const Vector &v){

int \*res = new int[k];

for(int i = 0; i < k; i++){

res[i] = a[i];

}

return res;

}

**Vector**(const Vector &v){

a = new int[v.k];

for(int i = 0; i < v.k; i++) a[i] = v.a[i];

k = v.k;

}

Vector operator+(Vector &v){

Vector res;

int k1 = getK()+v.getK();

int k2 = getK();

res.setK(k1);

int \*a1 = getA();

int \*a2 = v.getA();

for (int i = 0; i < k1;i++){

if(i < k2){

res[i] = a1[i];}

else res[i] = a2[i-k2];

}

return res;

}

int **getK**(){return k;}

int\* **getA**(){return a;}

void **setK**(int \_k) {

int \*buf = new int[k];

for(int i = 0; i < k;i++) buf[i] = a[i];

a = new int[\_k];

for(int i = 0; i< k; i++) a[i] = buf[i];

this->k = \_k;

}

int& operator[] (int i){

return a[i];

}

friend void operator <<(ostream &out, Vector &v){

int k = v.getK();

int \*a = v.getA();

for(int i = 0; i < k; i++)out << a[i] << " ";

}

friend void operator>>(istream &in, Vector &v){

cout << "number of elements : ";

int k;

in >> k;

v.setK(k);

int \*a =v.getA();

for(int i = 0; i < k; i++){

in >> a[i];

}

}

};

int **main**(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

int mas[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3, 4, 2, 2, 3, 3, 2, 1, 2, 3,4 };

Vector \*b = new Vector(mas, sizeof (mas)/sizeof (\*mas));

Vector c(mas, sizeof (mas)/sizeof (\*mas));

Vector \*d = new Vector(\*b+c);

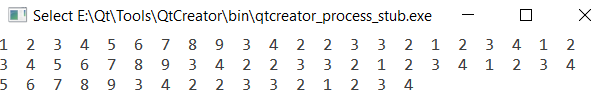
delete d;

cout << \*d;

cout << \*b;

delete b;

return a.exec();

}

**Висновок**: ознайомився зі способами перевантаження операцій та навчився їх використовувати при роботі з об’єктами.

# ЛАБОРАТОРНА РОБОТА №4

## **Тема:** Успадковування класів

## **Мета:** ознайомитись зі способами та механізмами успадкування класів та навчитись використовувати їх для побудови об’єктно-орієнтованих програм.

**Завдання 1.**

#include <QCoreApplication>

#include <iostream>

#include <string.h>

using namespace std;

class **publication**{

string name;

int price;

public:

string **getName**(){return name;}

int **getPrice**(){return price;}

void **setName**(string n){name = n;}

void **setPrice**(int p){price = p;}

void **getData**(){

cout << "Name = "+name+" price = " << price << endl;

}

void **setData**(){

cout << "name: ";

cin >> name ;

cout << endl << "price: ";

cin >> price;

cout << endl;

}

};

class **book**: public publication{

int amountPages;

public:

void **setAmountPages**(int a){

amountPages = a;

}

int **getAmountPages**(){

return amountPages;

}

void **getData**(){

publication::getData();

cout << "Amount Pages = " << amountPages << endl;

}

void **setData**(){

publication::setData();

cout << "Amount Pages :" ;

cin >> amountPages ;

cout << endl;

}

};

class **type**: public publication{

float time;

public:

void **setTime**(float t){

time = t;

}

void **getData**(){

publication::getData();

cout << "Time = " << time << endl;

}

void **setData**(){

publication::setData();

cout << "Time: ";

cin >> time;

cout << endl;

}

float **getTime**(){

return time;

}

};

int **main**(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

publication p;

book b;

type t;

p.setData();

b.setData();

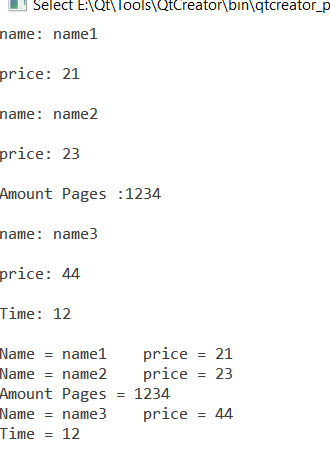
t.setData();

p.getData();

b.getData();

t.getData();

return a.exec();

}

**Завдання 2.**

#include <QCoreApplication>

#include <iostream>

using namespace std;

class **sales**{

float mounth\_sales[3];

public:

void **setData**(){

cout << "mounth sales: " << endl;

for(int i = 0; i < 3; i++){

cout << i+1 << " ";

cin >> mounth\_sales[i];

cout << endl;

}

}

void **getData**(){

cout << "mounth sales :" << endl;

for(int i = 0; i < 3; i++){

cout << i+1 << " " << mounth\_sales[i] << endl;

}

}

};

class **publication**{

string name;

int price;

public:

string **getName**(){return name;}

int **getPrice**(){return price;}

void **setName**(string n){name = n;}

void **setPrice**(int p){price = p;}

void **getData**(){

cout << "Name = "+name+" price = " << price << endl;

}

void **setData**(){

cout << "name: ";

cin >> name ;

cout << endl << "price: ";

cin >> price;

cout << endl;

}

};

class **book**:public publication, public sales {

int amountPages;

public:

void **setAmountPages**(int a){

amountPages = a;

}

int **getAmountPages**(){

return amountPages;

}

void **getData**(){

publication::getData();

sales::getData();

cout << "Amount Pages = " << amountPages << endl;

}

void **setData**(){

publication::setData();

sales::setData();

cout << "Amount Pages :" ;

cin >> amountPages ;

cout << endl;

}

};

class **type**:public publication, public sales{

float time;

public:

void **setTime**(float t){

time = t;

}

void **getData**(){

publication::getData();

sales::getData();

cout << "Time = " << time << endl;

}

void **setData**(){

publication::setData();

sales::setData();

cout << "Time: ";

cin >> time;

cout << endl;

}

float **getTime**(){

return time;

}

};

int **main**(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

book b;

b.setData();

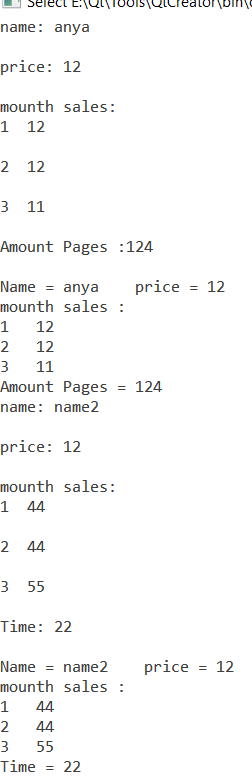
b.getData();

type t;

t.setData();

t.getData();

return a.exec();

}

**Завдання 3.**

#include <QCoreApplication>

#include <iostream>

using namespace std;

class **Pair**{

int fistNumber;

int secondNumber;

public :

**Pair**(int a = 0, int b = 0){

fistNumber = a;

secondNumber = b;

}

int **getFistNumber**(){

return fistNumber;

}

int **getsSecondNumber**(){

return secondNumber;

}

void **setFistNumber**(int f){

fistNumber = f;

}

void **setSecondNumber**(int s){

secondNumber = s;

}

friend void operator>>(iostream &input, Pair &p){

cout << "Fist Number: ";

input >> p.fistNumber;

cout << "Second Number: ";

input >> p.secondNumber;

cout << endl;

}

friend void operator<<(iostream &output, Pair &p){

output << "FistNumber = " << p.fistNumber << endl << "Secoond Number = " << p.secondNumber << endl;

}

bool operator<(Pair &p){

if(fistNumber < p.fistNumber)return true;

else if (fistNumber == p.fistNumber){

if( secondNumber < p.secondNumber)return true;

return false;

}

return false;

}

bool operator>(Pair &p){

if (\*this == p )return false;

return !(\*this < p);

}

bool operator==(Pair &p){

if (fistNumber == p.fistNumber && secondNumber == p.secondNumber)return true;

}

};

class **fraction**:public Pair{

};

int **main**(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

Pair b(9, 3), c(7, 8);

if(b < c){

cout << "b < c" << endl;

}

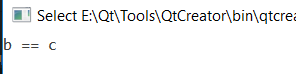
else if( b== c){

cout << "b == c" << endl;

}

else cout << "b > c" << endl;

return a.exec();

}

## **Висновок**: ознайомився зі способами та механізмами успадкування класів та навчився використовувати їх для побудови об’єктно-орієнтованих програм.

**Лабораторна робота №5**

**Тема**: Віртуальні функції та поліморфізм

**Мета**: практично ознайомитись з поняттям поліморфізму, його застосуванням та вивчити мехнізм його реалізації за допомогою віртуальних функцій.

**Завдання 1:**

#include <iostream>

#include <string>

using namespace std;

class **publication**{

string name;

int price;

public:

string **getName**(){return name;}

int **getPrice**(){return price;}

void **setName**(string n){name = n;}

void **setPrice**(int p){price = p;}

virtual void ***getData***(){

cout << "Name = "+name+" price = " << price << endl;

}

virtual void ***setData***(){

cout << "name: ";

cin >> name ;

cout << endl << "price: ";

cin >> price;

cout << endl;

}

};

class **book**: public publication{

int amountPages;

public:

void **setAmountPages**(int a){

amountPages = a;

}

int **getAmountPages**(){

return amountPages;

}

void ***getData***() override{

publication::getData();

cout << "Amount Pages = " << amountPages << endl;

}

void ***setData***() override{

publication::setData();

cout << "Amount Pages :" ;

cin >> amountPages ;

cout << endl;

}

};

class **type**: public publication{

float time;

public:

void **setTime**(float t){

time = t;

}

void ***getData***() override{

publication::getData();

cout << "Time = " << time << endl;

}

void ***setData***() override{

publication::setData();

cout << "Time: ";

cin >> time;

cout << endl;

}

float **getTime**(){

return time;

}

};

int **main**(int argc, char \*argv[])

{

publication\* a[] = {new book, new book, new type};

for(int i = 0; i < 3; i++){

a[i]->*setData*();

}

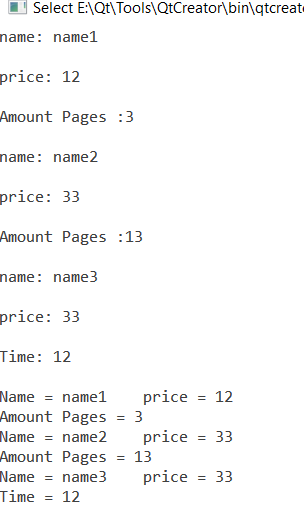
for(int i = 0; i < 3; i++){

a[i]->*getData*();

}

return 0;

}



**Завдання 2:**

#include <iostream>

#include <string>

using namespace std;

class **publication**{

string name;

int price;

public:

string **getName**(){return name;}

int **getPrice**(){return price;}

void **setName**(string n){name = n;}

void **setPrice**(int p){price = p;}

virtual void ***getData***(){

cout << "Name = "+name+" price = " << price << endl;

}

virtual void ***setData***(){

cout << "name: ";

cin >> name ;

cout << endl << "price: ";

cin >> price;

cout << endl;

}

};

class **book**: public publication{

int amountPages;

public:

bool **isOversize**(){

if(amountPages > 800){

cout <<"Перевищення розміру!"<<endl;

return true;

}

return false;

}

void **setAmountPages**(int a){

amountPages = a;

}

int **getAmountPages**(){

return amountPages;

}

void ***getData***() override{

publication::getData();

cout << "Amount Pages = " << amountPages << endl;

}

void ***setData***() override{

publication::setData();

cout << "Amount Pages :" ;

cin >> amountPages ;

cout << endl;

}

};

class **type**: public publication{

float time;

public:

bool **isOversize**(){

if(time > 90){

cout <<"Size Overload!"<<endl;

return true;

}

return false;

}

void **setTime**(float t){

time = t;

}

void ***getData***() override{

publication::getData();

cout << "Time = " << time << endl;

}

void ***setData***() override{

publication::setData();

do{

cout << "Time: ";

cin >> time;

}

while(isOversize());

cout << endl;

}

float **getTime**(){

return time;

}

};

int **main**(int argc, char \*argv[])

{

publication\* a[] = {new book, new book, new type};

for(int i = 0; i < 3; i++){

a[i]->*setData*();

}

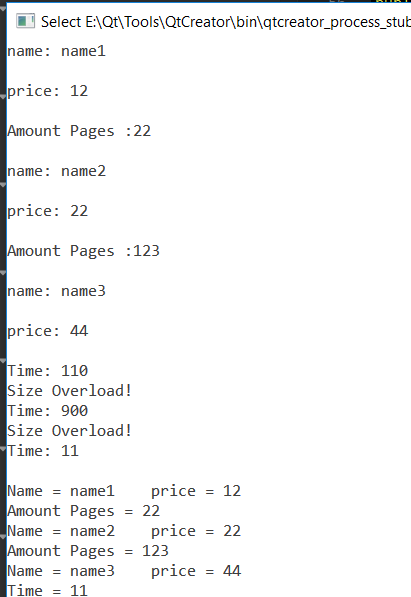
for(int i = 0; i < 3; i++){

a[i]->*getData*();

}

return 0;

}



**Завдання 3:**

**Лабораторна робота 6**

**Тема:** композиція об’єктів в ООП

**Мета:** ознайомитись зі способами та механізмами.

Завдання 2:

А) #include <iostream>

#include <student.h>

using namespace std;

int **main**()

{

Student a, b("b", "b", "b"), \*c = new Student("c", "c", "c");

cout << a;

cout << b;

cout << \*c;

return 0;

}

Б) #include "student.h"

#include <string>

#include <iostream>

Student::**Student**()

{

std::cout << "name :";

std::cin >> name ;

std::cout << std::endl << "surname : ";

std::cin >> surname ;

std::cout << std::endl << "lastname : ";

std::cin >> lastname;

}

Student::**Student**(std::string \_name, std::string \_surname, std::string \_lastname):name(\_name), surname(\_surname), lastname(\_lastname){};

В) #ifndef STUDENT\_H

#define STUDENT\_H

#include <iostream>

#include <string>

class **Student**

{

std::string name;

std::string surname;

std::string lastname;

public:

**Student**() ;

**Student**(std::string, std::string, std::string);

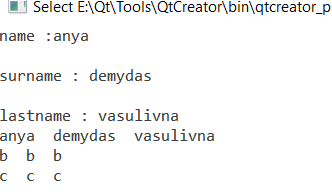
friend void operator<<(std::ostream &out, Student s){

out << s.name << " "<< s.surname<< " "<< s.lastname <<'\n' ;

}

};

#endif // STUDENT\_H



**Завдання 3:**

А) #include "grup.h"

Grup::**Grup**()

{

cout << "name grup : ";

cin >> name;

cout << endl << "specialty : ";

cin >> specialty;

int numberStudents;

cout <<endl<< "number Students: ";

cin >> numberStudents;

for(int i = 0; i < numberStudents; i++){

students.push\_back(new Student);

}

}

Б) #include <iostream>

#include <grup.h>

using namespace std;

int **main**()

{

Grup g;

cout << g;

return 0;

}

В) #include "student.h"

#include <string>

#include <iostream>

Student::**Student**()

{

std::cout << "name :";

std::cin >> name ;

std::cout << std::endl << "surname : ";

std::cin >> surname ;

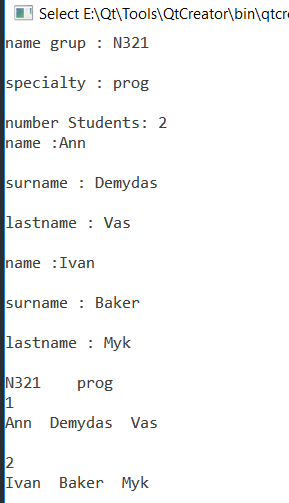
std::cout << std::endl << "lastname : ";

std::cin >> lastname;

std::cout << std::endl;

}

Student::**Student**(std::string \_name, std::string \_surname, std::string \_lastname):name(\_name), surname(\_surname), lastname(\_lastname){};



**Завдання 4:**

А) #include "facultet.h"

Facultet::**Facultet**()

{

cout << "Faculty name : ";

cin >> name ;

int numberGrups;

cout << endl << "number grups : ";

cin >>numberGrups;

for(int i = 0; i < numberGrups; i++){

grups.push\_back(Grup());

}

cout << endl;

}

Б) #include "grup.h"

Grup::**Grup**()

{

cout << "name grup : ";

cin >> name;

cout << endl << "specialty : ";

cin >> specialty;

int numberStudents;

cout <<endl<< "number Students: ";

cin >> numberStudents;

for(int i = 0; i < numberStudents; i++){

students.push\_back(new Student);

}

}

В) #include <iostream>

using namespace std;

int **main**()

{

cout << "Hello World!" << endl;

return 0;

}

Г) #include "student.h"

#include <string>

#include <iostream>

Student::**Student**()

{

std::cout << "name :";

std::cin >> name ;

std::cout << std::endl << "surname : ";

std::cin >> surname ;

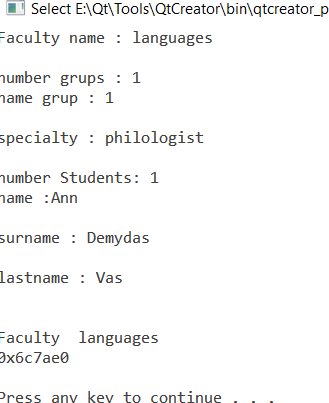
std::cout << std::endl << "lastname : ";

std::cin >> lastname;

std::cout << std::endl;

}

Student::**Student**(std::string \_name, std::string \_surname, std::string \_lastname):name(\_name), surname(\_surname), lastname(\_lastname){};



**Висновок**: ознайомився зі способами та механізмами.