**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**()

{

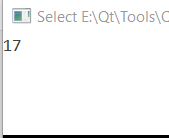
Int a = 9, b = 8;

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();

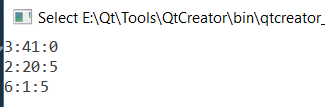
time t3;

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++) {

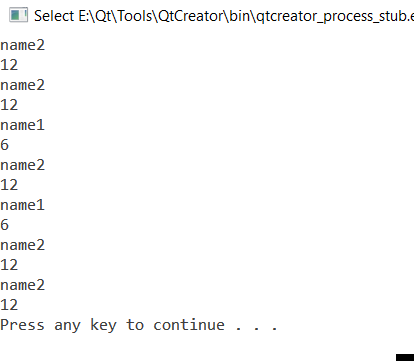
mas[i].Print();

}

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);

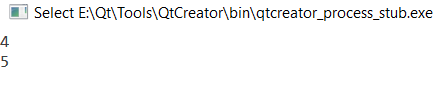
e.Print(4);

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);

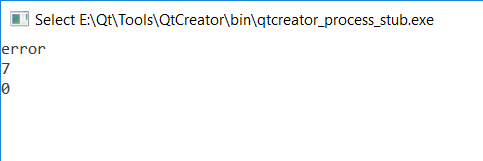
Int c(4);

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);

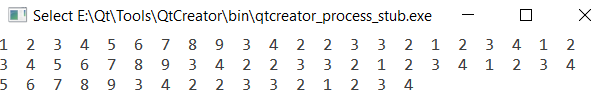
delete d;

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();

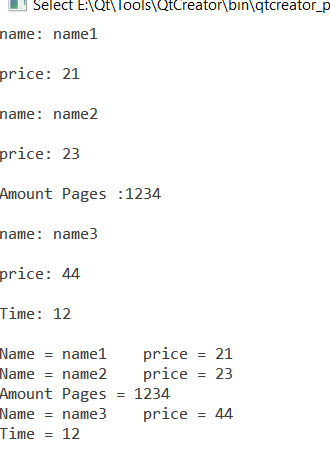
t.setData();

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();

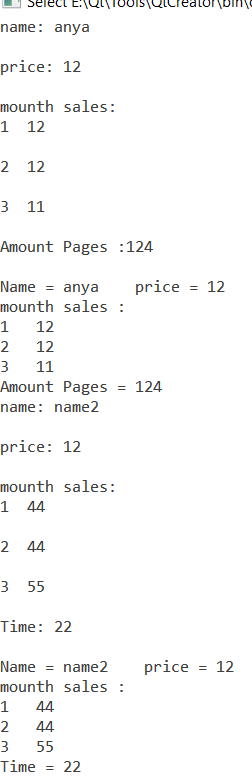
b.getData();

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;

}

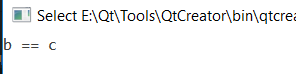
else if( b== c){

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*();

}

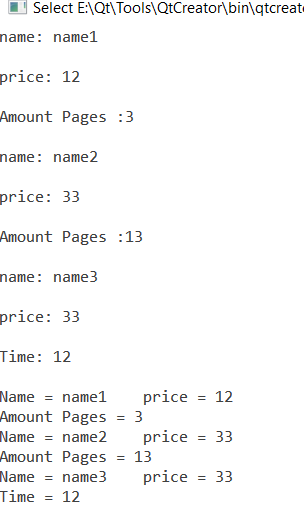
for(int i = 0; i < 3; i++){

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*();

}

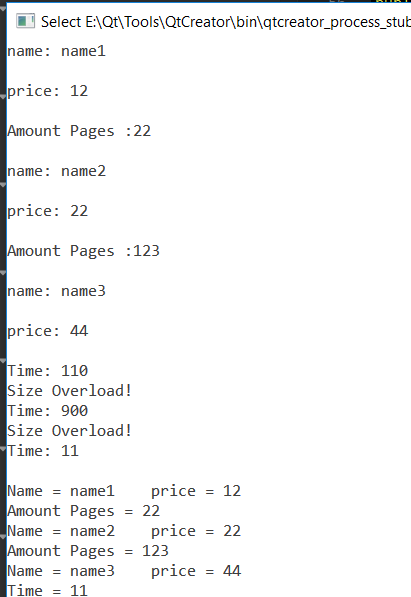
for(int i = 0; i < 3; i++){

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);

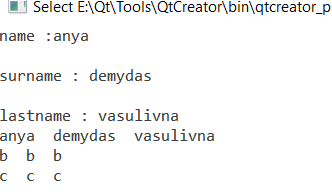
friend void operator<<(std::ostream &out, Student s){

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 ;

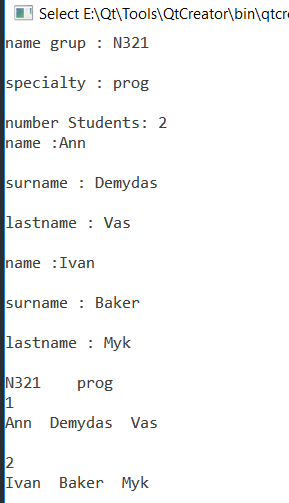
std::cout << std::endl << "lastname : ";

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 ;

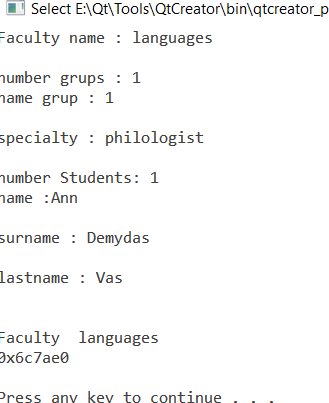
std::cout << std::endl << "lastname : ";

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.

***ЛАБОРАТОРНА РОБОТА №7***

**Тема:** Шаблони функцій і класів  
**Мета:** Навчитись створювати і використовувати шаблонні функції і класи

***ПОРЯДОК ВИКОНАННЯ РОБОТИ***

**Завдання 1.** Напишіть шаблон функції, що повертає середнє арифметичне всіх елементів масиву. Аргументами функції повинні бути ім'я і розмір масиву (типу int). У функції main() перевірте роботу шаблонної функції з масивами типу int, long, double і char.

**Завдання 2.** Створіть функцію аmaх(), що повертає значення максимального елемента масиву. Аргументами функції повинні бути адреса і розмір масиву. Зробіть з функції шаблон, щоб вона могла працювати з масивом будь-якого числового типу. Напишіть функцію main(), в якій перевірите роботу функції з різними типами масивів.

**Завдання 3.** Створіть шаблонний клас, який міститиме як атрибут - масив будь-якого числового типу. Розмір масиву необхідно визначати параметром конструктора класу. З допомогою методів класу потрібно:

• заповнювати масив;

• виводити значення масиву на екран;

• визначати і вивести середнє арифметичне всіх елементів масиву;

• визначати і вивести максимальний елемент масиву. Напишіть функцію main(), в якій перевірите роботу класу з різними вбудованими типами даних.

**Завдання 4.** Створити шаблонний клас – одновимірний масив. Тип елементів масиву визначається параметром шаблона. Передбачити функції для виконання таких перевантажених операцій: [] (з контролем діапазону індексу), = , <>.

Код програми:

Завдання 1:

#include <iostream>

using namespace std;

template <typename T>

T getMiddle(T t[], int length){

T sum = t[0];

for(int i = 1; i < length; i++){

sum += t[i];

}

return sum/length;

}

int main()

{

int mas1[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

double mas2[] = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9};

char mas3[] = {'q', 't', 'd', 'h', 'i', 'x', 'u', 'd', 'p', 'q', '0', '0', '0'};

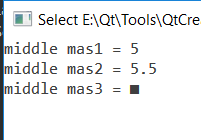
cout << "middle mas1 = " << getMiddle(mas1, sizeof(mas1)/sizeof(\*mas1)) << endl;

cout << "middle mas2 = " << getMiddle(mas2, sizeof(mas2)/sizeof(\*mas2)) << endl;

cout << "middle mas3 = " << getMiddle(mas3, sizeof(mas3)/sizeof(\*mas3)) << endl;

return 0;

}



Завдання 2:

#include <iostream>

using namespace std;

template <typename T>

T getMiddle(T t[], int length){

T sum = t[0];

for(int i = 1; i < length; i++){

sum += t[i];

}

return sum/length;

}

int main()

{

int mas1[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

double mas2[] = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9};

char mas3[] = {'q', 't', 'd', 'h', 'i', 'x', 'u', 'd', 'p', 'q', '0', '0', '0'};

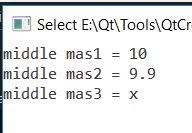
cout << "middle mas1 = " << getMiddle(mas1, sizeof(mas1)/sizeof(\*mas1)) << endl;

cout << "middle mas2 = " << getMiddle(mas2, sizeof(mas2)/sizeof(\*mas2)) << endl;

cout << "middle mas3 = " << getMiddle(mas3, sizeof(mas3)/sizeof(\*mas3)) << endl;

return 0;

}



Завдання 3:

#include <iostream>

using namespace std;

template <typename T>

class Array{

T \*arr;

int length;

public :

Array(int \_length){

length = \_length;

Init();

}

void Init(){

arr = new T[length];

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

cin >> arr[i];

}

}

void Print(){

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

cout <<arr[i] << " ";

}

cout << endl;

}

T getMax(){

T max = arr[0];

for(int i =1;i < length; i++){

if(max < arr[i]) max = arr[i];

}

return max;

}

T getMiddle(){

T sum = arr[0];

for(int i = 1; i < length; i++){

sum += arr[i];

}

return sum/length;

}

};

int main()

{

Array<int> a(12);

Array<char> b(10);

a.Print();

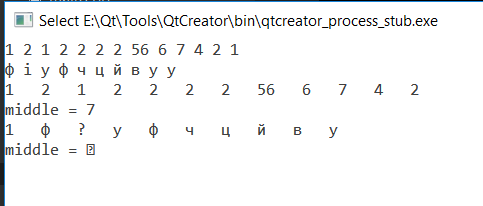
cout << "middle = " << a.getMiddle() << endl;

b.Print();

cout << "middle = " << b.getMiddle() << endl;

return 0;

}



Завдання 4:

#include <iostream>

#include <cstddef>

#include <string>

template <typename T>

class MyStack

{

private:

static const int N = 10;

T t[N];

std::size\_t counter;

public:

MyStack(): counter(0) {}

void push(const T &el)

{

if (counter < N) t[counter++] = el;

}

void pop()

{

if (counter > 0) counter--;

}

const std::size\_t size() const {return counter; }

bool empty() {return counter == 0; }

const T top() const

{

if (counter > 0) return t[counter - 1];

}

};

int main()

{

MyStack<int> s;

s.push (1);

s.push (2);

s.push (3);

s.push (4);

while (!s.empty())

{

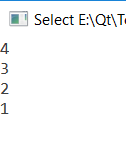
std::cout<< s.top()<< std::endl;

s.pop();

}

return 0;

}



***ЛАБОРАТОРНА РОБОТА №9***

**Тема:** Вивчення механізмів оброблення виняткових ситуацій  
**Мета:** Навчитись обробляти ситуації появлення виняткових ситуацій, вивчити механізми їх оброблення

***ПОРЯДОК ВИКОНАННЯ РОБОТИ***

**Завдання 1.** Додайте клас винятків до програми ARROVER таким чином, щоб індекси, що виходять за межі масиву, викликали генерацію винятку. Блокпастка catch може виводити користувачеві повідомлення про помилку.

#include <iostream>

using namespace std;

#include <process.h>

const int LIMIT = 100;

class ARROVER {

};

class safearay {

private:

int arr[LIMIT];

public:

int& operator [](int n)

{

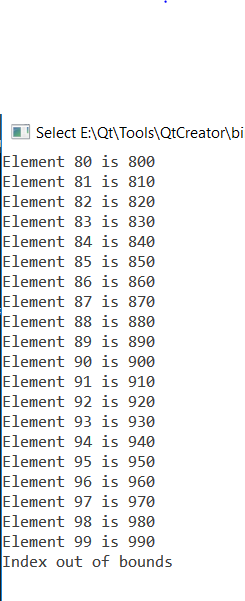
try{

if( n< 0 || n>=LIMIT ) throw ARROVER();

return arr[n];

}catch(ARROVER e){

cout << "Index out of bounds" << endl;

 exit(1);

}

}

};

int main()

{ safearay sa1;

for(int j=0; j<LIMIT; j++)

sa1[j] = j\*10; //\*left\* side of equal sign

for(int j=0; j<LIMIT; j++) //display elements

{

int temp = sa1[j]; //\*right\* side of equal sign

cout << "Element " << j << " is " << temp << endl;

}

cout << sa1[123] << endl;

cout << sa1[1] << endl;

return 0;

}

**Завдання 2.** Модифікуйте текст програми, розробленої згідно індивідуального завдання лабораторної роботи №3, добавивши до нього (в місця можливого виникнення помилок) процедури оброблення виняткових ситуацій, які будуть генерувати об’єкти класу, який міститиме такі атрибути: місце виникнення помилки; значення, яке призвело до помилки; параметризований конструктор; перевизначену операції виводу, яка виводитиме значення на екран (або зберігатиме виняткову ситуацію у файл).

Код програми:

#include <iostream>

using namespace std;

#include <process.h>

const int LIMIT = 100;

class ARROVER {

int n ;

public:

ARROVER(int n ){

cout <<endl << "n > " << LIMIT << endl;

}

};

class safearay {

private:

int arr[LIMIT];

public:

int& operator [](int n)

{

try{

if( n< 0 || n>=LIMIT ) throw ARROVER(n);

return arr[n];

}catch(ARROVER e){

cout << "Index out of bounds" << endl;

exit(1);

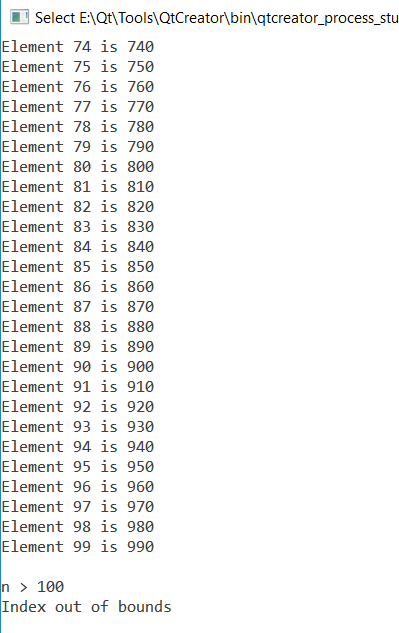
}

}

};

int main()

{ safearay sa1;



for(int j=0; j<LIMIT; j++)

sa1[j] = j\*10; //\*left\* side of equal sign

for(int j=0; j<LIMIT; j++) //display elements

{

int temp = sa1[j]; //\*right\* side of equal sign

cout << "Element " << j << " is " << temp << endl;

}

cout << sa1[123] << endl;

cout << sa1[1] << endl;

return 0;

}

**Висновок:** Навчився обробляти ситуації появлення виняткових ситуацій, вивчив механізми їх оброблення.

***ЛАБОРАТОРНА РОБОТА №10***

**Тема:** Робота з файлами  
**Мета:** Навчитись зчитувати і записувати данні у текстові і двійкові файли за допомогою файлових потоків

***ПОРЯДОК ВИКОНАННЯ РОБОТИ***

**Завдання 1.** Написати програму, яка записуватиме ваші дані у текстовий файл «myfile.txt» (П.І.П., вік, номер телефону, дата народження, місце навчання). Дані для запису можна або зчитувати з консолі при виконанні програми або прописати у коді програми.

**Завдання 2.** Написати програму, яка здійснить читання файла «myfile.txt» у відповідні локальні змінні і виведе їх значення на екран.

Код програми:

#include <QCoreApplication>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

const string fileName = "myfile.txt";

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

{

QCoreApplication a(argc, argv);

//---------------------1-----------------------

ofstream outFile(fileName);

if(outFile.is\_open())

{

string strToWrite = "Pelishko Maksym Ruslanovych 17 0689479273 08.01.2001 TK\_TNTU";

outFile.write(strToWrite.c\_str(), strToWrite.size());

outFile.close();

}else cout << "Can't create or open file!";

//---------------------2-----------------------

ifstream inFile(fileName);

if(inFile.is\_open())

{

string surname, name, fatherName, birthday, educationPlace;

int age, phoneNumber;

inFile >> surname >> name >> fatherName >> age >> phoneNumber >> birthday >> educationPlace;

inFile.close();

cout << "Person - " << surname << " " << name << " " << fatherName << endl;

cout << "Age - " << age << ", birthday - " << birthday << endl;

cout << "Phone number - " << phoneNumber << endl;

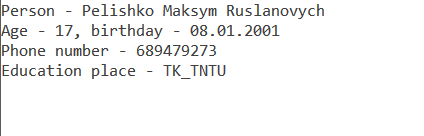
cout << "Education place - " << educationPlace << endl;

}else cout << "Can't open file!";

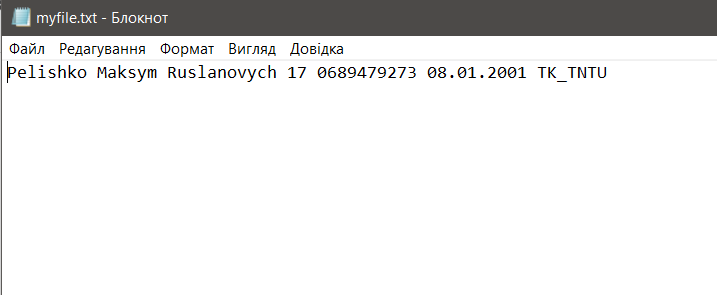
return a.exec();

}

Вивід програми:



Вміст файлу “myfile.txt”



**Завдання 3.** Створити клас Person, який міститиме атрибутами ті дані, що записувались у завданні №2. Дані повинні зчитуватись із клавіатури при створенні об’єкта. Створити об’єкт ME і записати його у файл. (Перевизначити оператор запису у потік, який запише дані у файл).

#include <iostream>

#include <fstream>

#include <string.h>

using namespace std;

class Person{

string name;

string surname;

int age;

string phone;

string education\_palse;

public:

Person(){

cout << "name, surname, age, phone, education\_plase" << endl;

cin >> name ;

cin >> surname;

cin >> age;

cin>> phone;

cin>> education\_palse;

cout << endl;

}

friend void operator<<(ofstream &out, Person &p){

out << p.name << " " << p.surname <<" " << p.age <<" " << p.phone <<" " << p.education\_palse << endl;

}

};

int main()

{

ofstream myfile;

Person p;

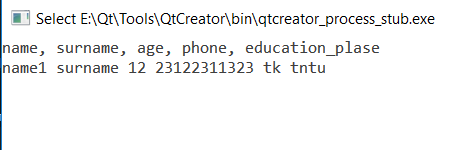
myfile.open ("myfile.txt");

myfile << p;

myfile.close();

return 0;

}



**Завдання 4.** Використовуючи клас Person зчитайте дані, що були записані у попередньому завданні. (Потрібно перевизначити оператор зчитування з потоку, який зчитуватиме дані із файлу).

#include <iostream>

#include <fstream>

#include <string.h>

using namespace std;

class Person{

string name;

string surname;

int age;

string phone;

string education\_palse;

public:

Person(){

}

friend void operator>>(ifstream &out, Person &p){

out >> p.name ;

out >> p.surname;

out >> p.age;

out >> p.phone;

out >> p.education\_palse;

cout << endl;

}

friend void operator<<(ostream &out, Person &p){

out << p.name << " " << p.surname <<" " << p.age <<" " << p.phone <<" " << p.education\_palse << endl;

}

};

int main()

{

ifstream myfile;

Person p;

myfile.open ("myfile.txt");

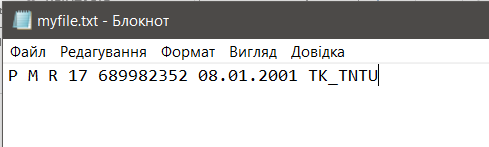
myfile >> p;

cout << p;

myfile.close();

return 0;

}



**Висновок:** Навчився зчитувати і записувати данні у текстові і двійкові файли за допомогою файлових потоків.

***ЛАБОРАТОРНА РОБОТА №11***

**Тема:** Дослідження контейнерних класів бібліотеки STL  
**Мета:** Дослідити контейнерні класи vector та list бібліотеки STL, набути навичок їх використання

***ПОРЯДОК ВИКОНАННЯ РОБОТИ***

**Завдання 1.** З допомогою контейнера std::vector створити вектор значень типу char, в який записати український алфавіт (малими літерами).

**Завдання 2.** Усі голосні букви у векторі замінити прописними (за допомогою циклу).

**Завдання 3.** Вивести вміст вектора на екран.

**Завдання 4.** З допомогою контейнера list створити список значень типу char, в який записати по буквах своє прізвище. Вивести вміст списку на екран.

**Завдання 5.** Відсортувати список (від А до Я). У консоль вивести повідомлення: «Список сортується!».

**Завдання 6.** Вивести вміст списку на екран.

**Завдання 7.** Створити вектор об’єктів класу, створеного згідно індивідуального завдання №1 в лабораторній роботі №2. Заповнити вектор десятьма об’єктами.

**Завдання 8.** Здійснити вивід значень об’єктів за допомогою індексу вектора.

**Завдання 9.** Здійснити вивід значень об’єктів за допомогою ітераторів.

**Завдання 10\***. З допомогою контейнера list створити список студентів групи. В контейнер поміщати об’єкти класу, в якому представлені такі атрибути: прізвище, ім’я, по батькові, вік, номер телефону, а також реалізовані сетери даних атрибутів, і перевизначена операція виводу, яка буде виводити значення атрибутів на екран.

Код програми:

#include <QCoreApplication>

#include <iostream>

#include <string>

using namespace std;

class Furniture

{

char \*Room = new char;

int Weight;

public:

Furniture(char\* room, int weight)

{

SetRoom(room);

Weight = weight;

}

Furniture():Furniture("Unknown", 0) {}

Furniture(const Furniture& furn)

{

//cout << "copy" << endl;

Weight = furn.Weight;

SetRoom(furn.Room);

}

~Furniture() {/\*delete Room;\*/ }

void SetRoom(const char\* room)

{

delete Room;

Room = new char[40];

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

Room[i] = room[i];

}

char\* GetRoom(){return Room;}

void SetWeight(int val) { Weight = val; }

int GetWeight() { return Weight; }

void Print() { cout << "Mass: " << Weight << ", room: " << Room << endl;

}

void Input() {

cout << "Write mass: ";

cin >> Weight;

cout << "Room: ";

cin >> Room;

}

};

class Student

{

string name,surname, fatherName, phoneNumber;

int age;

public:

Student(string name, string surname, string fatherName, string phoneNumber, int age)

{

this->name = name;

this->surname = surname;

this->fatherName = fatherName;

this->phoneNumber = phoneNumber;

this->age = age;

}

string GetName(){return name;}

string GetSurname(){return surname;}

string GetFatherName(){return fatherName;}

string GetPhoneNumber(){return phoneNumber;}

int GetAge(){return age;}

void Print()

{

cout << "Student - " << name << " " << surname << " " << fatherName << endl << "Phone number - " << phoneNumber << endl <<

"Age - " << age << endl;

}

};

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

{

QCoreApplication a(argc, argv);

//--------------1---------------------

setlocale(0, "RUS");

vector<wchar\_t> alphabet; //= {L'а', L'б', L'в', L'г', L'u\1169', L'д'};

for(int i = 0; i < 33; i++)

{

alphabet.push\_back((wchar\_t)((int)(L'а')+i));

wcout << alphabet[i] << " ";

}

cout << endl;

//--------------2---------------------

vector<char> v;

for(auto i: "abcdefgigklnoprstuvwxyz")

v.push\_back(i);

for(unsigned int i = 0 ;i < v.size(); i++)

if(QString("aioyeu").contains(v[i]))

v[i] = (char)(v[i]-32);

for(auto i : v)

cout << i << " " ;

cout << endl;

//--------------3---------------------

vector<int> vect = {1,2,4,5,6,7,8};

for(unsigned int i = 0; i < vect.size(); i++)

cout << vect.at(i) << " ";

cout << endl;

//--------------4---------------------

list<char> surname;

QString n = "Pelishko";

for(auto i:n.toStdString())

surname.push\_back(i);

for(auto i:n.toStdString())

cout << i;

cout << endl;

//--------------5---------------------

vector<char> alphabet1;

for(int i = 25; i >= 0; i--)

alphabet1.push\_back((char)(i+97));

sort(alphabet1.begin(), alphabet1.end());

cout << "Array is sorting" << endl;

//--------------6---------------------

for(unsigned int i = 0; i < alphabet1.size(); i++)

cout << alphabet1[i] << " ";

cout << endl;

//--------------7---------------------

vector<Furniture> vect1;

for(int i = 0; i < 10; i++) vect1.push\_back(Furniture());

//--------------8---------------------

for(unsigned int i = 0; i < vect1.size(); i++)

vect1[i].Print();

cout << endl;

//--------------9---------------------

vector<Furniture>::iterator iter;

for(iter = vect1.begin(); iter != vect1.end(); iter++)

iter->Print();

cout << endl;

//--------------10--------------------

list<Student> listStud;

list<Student>::iterator iter1;

for(int i = 0; i < 2; i++)

listStud.push\_back(Student("q", "w", "e", "a", 1234));

for(iter1 = listStud.begin(); iter1 != listStud.end(); iter1++)

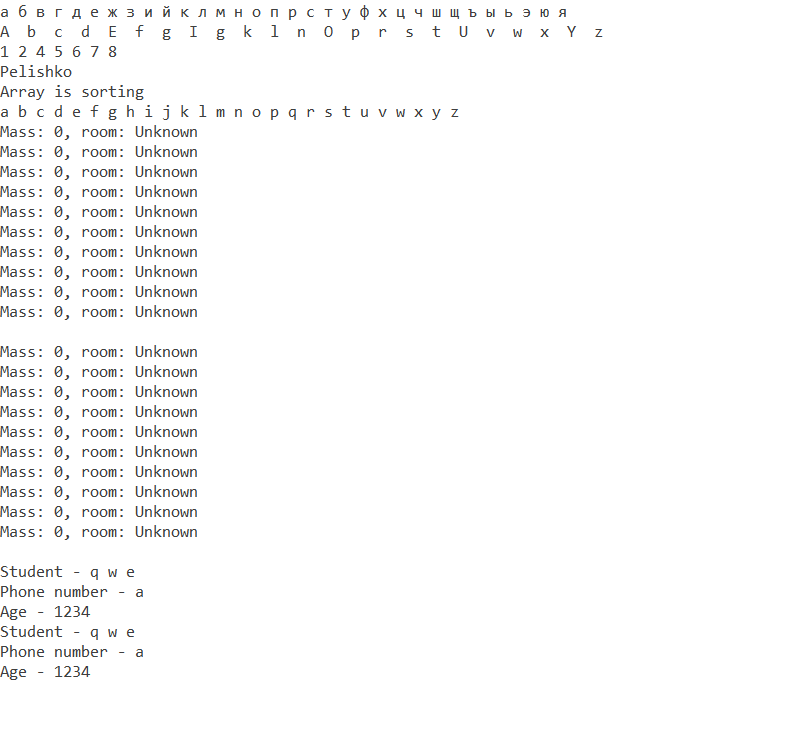
iter1->Print();

cout << endl;

return a.exec();

}

Вивід програми:



**Висновок:** Дослідив контейнерні класи vector та list бібліотеки STL, набув навичок їх використання.