**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **ПЗ**



### ЗВІТ

До лабораторної роботи № 7

**На тему:** *“* *Робота з динамічною пам’яттю”*

**З дисципліни:** *“Об’єктно-орієнтоване програмування”*

**Лектор:**

доцент каф.ПЗ

Коротєєва Т. О.

**Виконав:**

ст. гр. ПЗ-16

Шеремета А.І.

**Прийняв:**

асист. каф. ПЗ

Дивак І.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_ .

Львів – 2022

**Тема роботи:** Робота з динамічною пам’яттю.

**Мета** **роботи:** Навчитися виділяти місце під об’єкти динамічно. Навчитися створювати та використовувати конструктор копіювання, перевантажувати оператор присвоєння. Ознайомитися з принципами створення та функціонування деструкторів.

**Теоретичні відомості**

На вашому комп’ютері є пам’ять (можливо, більша її частина), яка доступна для використання програмами. При запуску вашої програми операційна система завантажує її в деяку частину цієї пам’яті. І ця пам’ять, яка використовується вашою програмою, розділена на кілька частин, кожна з яких виконує певне завдання. Одна частина містить ваш код, інша використовується для виконання звичайних операцій (відстеження функцій, створення і знищення глобальних і локальних змінних тощо). Ми поговоримо про це трохи пізніше. Тим не менш, велика частина доступної пам’яті комп’ютера просто знаходиться в очікуванні запитів на виділення від програм.

Коли ви динамічно виділяєте пам’ять, то ви просите операційну систему зарезервувати частину цієї пам’яті для використання вашою програмою. Якщо ОС може виконати цей запит, то повертається адреса цієї пам’яті назад в вашу програму. З цього моменту і надалі ваша програма зможе використовувати цю пам’ять, як тільки побажає. Коли ви вже виконали з цією пам’яттю все, що було необхідно, то її потрібно повернути назад в операційну систему для розподілу між іншими запитами на виділення.

Коли ви динамічно виділяєте змінну, то ви також можете її ініціалізувати за допомогою прямої ініціалізації або uniform-ініціалізації:

int \*ptr1 = new int (7); // використовуємо пряму ініціалізацію

int \*ptr2 = new int { 8 }; // використовуємо uniform-ініціалізацію

Коли вже все, що було потрібно, виконано з динамічно виділеною змінної — потрібно явно вказати для С++ звільнити цю пам’ять. Для змінних це виконується за допомогою оператора delete:

// Припустимо, що ptr раніше уже був виділений за допомогою оператора new

delete ptr; // повертаємо пам'ять, на яку вказував ptr, назад в операційну систему

ptr = 0; // робимо ptr нульовим вказівником

Оператор delete насправді нічого не видаляє. Він просто повертає пам’ять, яка була виділена раніше, назад в операційну систему. Потім операційна система може перепризначити цю пам’ять іншому додатку/програмі (або цій же програмі знову).

Хоча може здатися, що ми видаляємо змінну, але це не так! Змінна-вказівник має ту ж область видимості, що і раніше, і їй можна присвоїти нове значення, як і будь-якій іншій змінній.

Оператор new намагається вибрати та ініціалізувати об’єкт або масив об’єктів зазначеного типу чи заповнювача та повертає відповідний введений ненульовий покажчик на об’єкт (або на вихідний об’єкт масиву).

У разі невдачі new повертається нуль або виникає виняток.

Якщо new використовується виділення пам'яті для об'єкта класу C++, конструктор об'єкта викликається після виділення пам'яті.

**Лабораторне завдання**

1. Переглянути лістинг коду в прикладі. Пояснити вивід програми.
2. Створити клас відповідно до завдання (див. Додаток).
3. Розробити для класу конструктор за замовчуванням та декілька звичайних конструкторів. Реалізувати функції-члени відповідно до завдання (див. Додаток).
4. Створити конструктор копіювання.
5. Перевантажити операцію присвоєння.
6. Створити деструктор для вивільнення динамічно виділеної пам’яті.
7. Об'єкти класу розмістити в динамічній пам'яті.
8. Продемонструвати розроблені можливості класу завдяки створеному віконному застосуванню.
9. Оформити звіт до лабораторної роботи.

**Варіант 2**

Клас Array – одновимірний масив. Пам’ять під елементи масиву повинна виділятися динамічно.

Реалізувати такі функції члени:

* Знаходження максимального значення.
* Знаходження мінімального значення.
* Знаходження середнього арифметичного значення масиву.
* Сортування елементів масиву методом вибірки за спаданням.
* Сортування елементів масиву методом бульбашки за зростанням.
* Зміна розмірів масиву.

Перевантажити операції. При цьому вибір механізму перевантаження обрати самостійно (чи метод, чи дружня-функція):

* Додавання (почленне додавання елементів масиву)
* Віднімання (почленне віднімання елементів масиву)
* Множення на скаляр.
* Введення масиву з QTableWidget (>>)
* Виведення масиву у QTableWidget (<<)
* Введення масиву з QPlainTextEdit (>>)
* Виведення масиву у QPlainTextEdit (<<)
* Виведення масиву у Memo (<<)

Забезпечити можливість отримання значення елементу [i] подібно до доступу до елементів звичайного одновимірного масиву.

**Хід роботи**

1. Реалізовую програму:

**Файл mainwindow.cpp**

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

tabA = this->ui->tabArray;

Memo = this->ui->Memo;

tabA->setColumnCount(5);

connect(ui->pushButtonMax, SIGNAL(clicked(bool)), this, SLOT(pushButtonMaxClick()));

connect(ui->pushButtonMin, SIGNAL(clicked(bool)), this, SLOT(pushButtonMinClick()));

connect(ui->pushButtonAvg, SIGNAL(clicked(bool)), this, SLOT(pushButtonAvgClick()));

connect(ui->pushButtonChangeSize, SIGNAL(clicked(bool)), this, SLOT(pushButtonChangeSizeClick()));

connect(ui->pushButtonClear, SIGNAL(clicked(bool)), this, SLOT(pushButtonClearClick()));

connect(ui->pushButtonSortDown, SIGNAL(clicked(bool)), this, SLOT(pushButtonSortDownClick()));

connect(ui->pushButtonSortUp, SIGNAL(clicked(bool)), this, SLOT(pushButtonSortUpClick()));

connect(ui->pushButtonPlus, SIGNAL(clicked(bool)), this, SLOT(pushButtonPlusClick()));

connect(ui->pushButtonMinus, SIGNAL(clicked(bool)), this, SLOT(pushButtonMinusClick()));

connect(ui->pushButtonMultiply, SIGNAL(clicked(bool)), this, SLOT(pushButtonMultiplyClick()));

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::pushButtonMaxClick(){

Array A(tabA->columnCount());

A>>tabA;

ui->label\_Result->setText(QString::number(A.findmax()));

A<<\*Memo;

}

void MainWindow::pushButtonClearClick()

{

Array A(tabA->columnCount());

A.clear();

tabA->clear();

ui->tabArray->clear();

tabA->setColumnCount(5);

ui->tabArray->setGeometry(190, 60, 379, 81);

Memo->clear();

setCursor(Qt::ArrowCursor);

ui->newSize->clear();

ui->Number->clear();

}

void MainWindow::pushButtonAvgClick(){

Array A(tabA->columnCount());

A>>tabA;

ui->label\_Result->setText(QString::number(A.findavg()));

A<<\*Memo;

}

void MainWindow::pushButtonChangeSizeClick()

{

Array A(tabA->columnCount());

int newSize = ui->newSize->toPlainText().toInt();

if (newSize > 9){

ui->tabArray->setGeometry(60, 60, 700, 81);

}

else if(newSize == 4){

ui->tabArray->setGeometry(248, 60, 304, 81);

}

else if(newSize == 3){

ui->tabArray->setGeometry(285, 60, 229, 81);

}

else if(newSize == 2){

ui->tabArray->setGeometry(323, 60, 154, 81);

}

else if(newSize == 1){

ui->tabArray->setGeometry(359, 60, 81, 81);

QPixmap cursor\_pixmap = QPixmap(":/new/prefix1/cursor.png");

QCursor cursor\_default = QCursor(cursor\_pixmap, 112, 112);

QCursor::setPos(960, 540);

setCursor(cursor\_default);

}

else if (newSize <= 0){

QMessageBox about;

about.setInformativeText("G");

about.setText("Incorrect array size input");

about.addButton("Oke, I promise I will change it now", QMessageBox::AcceptRole);

about.setIconPixmap(QPixmap(":/new/prefix1/noarray.png"));

about.show();

about.exec();

}

else{

ui->tabArray->setGeometry(190, 60, 379, 81);

}

A.changesize(newSize);

tabA->setColumnCount(newSize);

}

void MainWindow::pushButtonMinClick(){

Array A(tabA->columnCount());

A>>tabA;

ui->label\_Result->setText(QString::number(A.findmin()));

A<<\*Memo;

}

void MainWindow::pushButtonSortDownClick()

{

Array A(tabA->columnCount());

A>>tabA;

A.sortdecrease();

ui->tabArray->clear();

A<<\*tabA;

A<<\*Memo;

}

void MainWindow::pushButtonSortUpClick(){

Array A(tabA->columnCount());

A>>tabA;

A.sortgrow();

ui->tabArray->clear();

A<<\*tabA;

A<<\*Memo;

}

void MainWindow::pushButtonPlusClick(){

Array A(tabA->columnCount());

A>>tabA;

A+=ui->Number->toPlainText().toInt();

A<<\*tabA;

A<<\*Memo;

}

void MainWindow::pushButtonMinusClick(){

Array A(tabA->columnCount());

A>>tabA;

A-=ui->Number->toPlainText().toInt();

A<<\*tabA;

A<<\*Memo;

}

void MainWindow::pushButtonMultiplyClick(){

Array A(tabA->columnCount());

A>>tabA;

A\*=ui->Number->toPlainText().toInt();

A<<\*tabA;

A<<\*Memo;

}

**Файл array.cpp**

#include "array.h"

Array::Array(){

size = 0;

arr = nullptr;

}

Array::Array(int r){

size = r;

arr = new int [size];

}

Array::Array(const Array &other){

this->size = other.size;

this->arr = new int[other.size];

for(int i = 0; i < other.size; i++){

this->arr[i] = other.arr[i] ;

}

}

Array::~Array(){

delete [] arr;

}

Array & Array::operator=(const Array &other){

if(this != &other){

this->size = other.size;

delete[] this->arr;

this->arr = new int[other.size];

for(int i = 0; i < other.size; i++){

this->arr[i] = other.arr[i];

}

}

return \*this;

}

void Array::operator>>(QTableWidget\* tabArray){

Array A(tabArray->columnCount());

for(int i = 0; i<tabArray->columnCount(); i++){

int element = tabArray->item(0, i)->text().toInt();

arr[i] = element;

}

}

void Array::operator<<(QTableWidget& tabArray){

tabArray.setColumnCount(size);

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

QTableWidgetItem \*newitem = new QTableWidgetItem;

newitem->setText(QString::number(arr[i]));

tabArray.setItem(0,i,newitem);

}

}

void Array::operator<<(QPlainTextEdit &Memo){

Memo.clear();

QString str = " ";

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

str += QString::number(arr[i]);

str.append(" ");

}

Memo.appendPlainText(str);

}

int & Array::operator[](int index){

return arr[index];

}

Array & Array::operator+=(int Number){

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

this->arr[i]+= Number;

}

return \*this;

}

Array & Array::operator-=(int Number){

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

this->arr[i]-= Number;

}

return \*this;

}

Array & Array::operator\*=(int scalar){

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

this->arr[i]\*= scalar;

}

return \*this;

}

int Array::findmax(){

int max = this->arr[0];

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

if(this->arr[i] > max)

max = this->arr[i];

}

return max;

}

void Array::changesize(int s){

// temp..

// delete arr[];

size = s;

}

double Array::findavg(){

double avg = 0;

int sum = 0;

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

int elem = arr[i];

sum += elem;

}

avg = sum/size;

return avg;

}

int Array::findmin(){

int min = this->arr[0];

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

if(this->arr[i] < min)

min = this->arr[i];

}

return min;

}

void Array::clear(){

size = 0;

delete[] arr;

arr = nullptr;

}

void Array::sortdecrease(){

// цикл отримує доступ до кожного елементу

for (int step = 0; step < (size-1); ++step) {

// перевіряє чи вже сортовано

int swapped = 0;

// цикл порівнює елементи

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

// порівнює 2 елементи масиву

if (arr[i] < arr[i + 1]) {

// заміна відбувається, якщо елементи

// розташовані не в належному порядку

int temp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = temp;

swapped = 1;

}

}

// не міняє елементи якщо масив вже сортований

// тому перестає порівнювати

if (swapped == 0)

break;

}

}

void Array::sortgrow(){

int i, j, temp;

for (i = 1; i < size; i++){

temp = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > temp){

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = temp;

}

}

**Файл mainwindow.h**

#include <QMainWindow>

#include <array.h>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void pushButtonMaxClick();

void pushButtonClearClick();

void pushButtonChangeSizeClick();

void pushButtonAvgClick();

void pushButtonMinClick();

void pushButtonSortDownClick();

void pushButtonSortUpClick();

void pushButtonPlusClick();

void pushButtonMinusClick();

void pushButtonMultiplyClick();

private:

Ui::MainWindow \*ui;

QTableWidget \*tabA;

QPlainTextEdit \*Memo;

};

#endif // MAINWINDOW\_H

**Файл array.h**

#ifndef ARRAY\_H

#define ARRAY\_H

#include <QTableWidget>

#include <QPlainTextEdit>

class Array

{

private:

int size;

int\* arr;

public:

Array();

Array(int r);

Array(const Array &other);

Array& operator=(const Array &other);

~Array();

int findmax();

int findmin();

double findavg();

void sortdecrease();

void sortgrow();

void changesize(int r);

int getSize();

void clear();

int & operator[](int index);

void operator>>(QTableWidget \*tableWidget);

void operator<<(QTableWidget &tabArray);

void operator<<(QPlainTextEdit &Memo);

Array& operator+=(int Number);

Array& operator-=(int Number);

Array& operator\*=(int scalar);

};

#endif // ARRAY\_H

**Виконання лабораторної роботи**

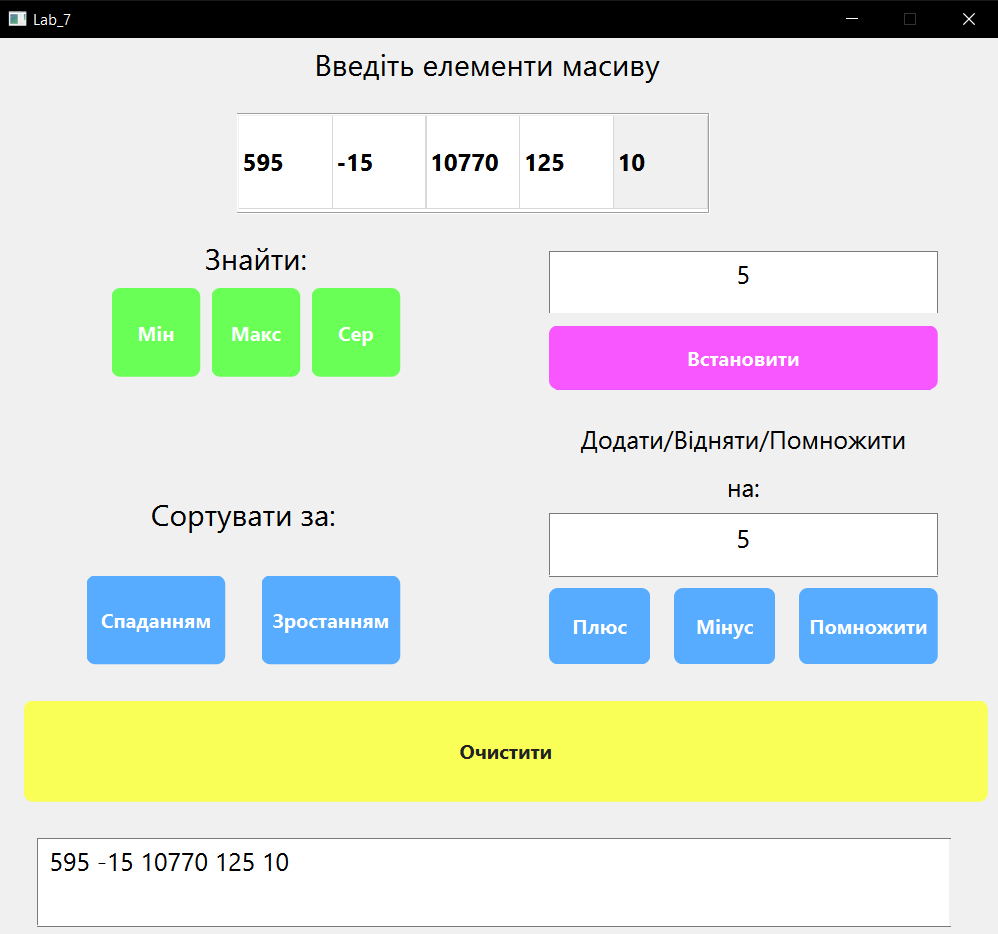


Рис. 1 Виконання програми

**Висновки**

На цій лабораторній роботі я навчився використовувати динамічну пам’ять для об’єктів класу, використовувати конструктор копіювання, перевантажувати оператор присвоєння. Ознайомився з принципами функціонування деструкторів.