

1. Постановка задачи

Важно учитывать при реализации:

- Необходимо выполнить разделение на `h` и `src` файлы для каждого класса. `h` файлы содержат определение, `src` файлы содержат реализацию. функция `main` обязана располагаться в отдельном `src` файле.
- Элемент очереди/стека/списка содержит данные и ссылку на предыдущий/следующий/предыдущий и следующий элемент. Элемент реализовать с помощью класса или структуры. В классе или структуре, реализующей элемент, необходимо поместить функции для установки и извлечения данных. То есть прямую обращения к данным и ссылке(-ам) быть не должно.
- Необходимо работать с динамическим выделением памяти. Важно выделять и всегда очищать выделенную память.
- Необходимо написать для каждого класса: конструктор с параметрами и без параметров, конструктор копирования, деструктор, необходимо продемонстрировать применение спецификатора `explicit`.
- Требуется реализовать методы/функции с аргументами по умолчанию.
- Реализовать пользовательское меню согласно заданию. В пользовательском меню обязательно предоставить пользователю возможность взаимодействовать со всеми настраиваемыми параметрами. То есть не должно существовать в программе числовых или буквенных констант, которые могут быть введены пользователем.
- Номер варианта соответствует номеру студента в списке. Каждому студенту необходимо выполнить оба задания.

Вариант 17.

- Задание 1 Унарная операция

Создать класс координаты (`x`, `y`, `z`). Перегрузить операторы: `!` умножает все координаты на `(-1)`, `++` (префиксная форма, дружественная функция) увеличивает все координаты на наименьшее значение среди `x`, `y`, `z`; `++` (постфиксная форма, дружественная функция) увеличивает одну из координат на пользовательское число; `--` (префиксная форма, метод) уменьшает все координаты на наибольшее значение среди `x`, `y`, `z`; `--` (постфиксная форма, метод) уменьшает одну из координат на пользовательское число.

- Задание 2 Бинарная операция

Создать объект "стек" с перегруженными операциями для работы с числами (половину как дружественные функции, половину как методы) `-`, `-=`, `/=`, `/`, `=`, `==`, `<`, `>`, `!`, ввод и вывод в поток. Размер стека определяется случайным образом (значение от 5 до 10), также случайным образом заполняется.

2. Формализация

2.1 Формализация задания №1

Исходя из требований будет создан класс `Point` с приватными полями `_x`, `_y`, `_z`, `_increment`, `_decrement`. Последние два поля представляют собой «пользовательское число». Для всех полей будут геттеры и сеттеры. Помимо конструкторов, деструкторов, функции вывода и перегрузки операторов будет функция `turnAll` с параметром по умолчанию – функция будет двигать вправо все координаты на шаг `step` по умолчанию равный 100.

2.2 Формализация задания №2

Исходя из требований будет создан класс `MyStack` со внутренним приватным классом `Node` и приватными полями `Node *head`, `*end` и длина `len`. Также будут реализованы внутренние `protected` функции `contain` и `remove`, позволяющие получить доступ к данным стека, не копируя его. Пусть операция вычитания для стэков удаляет из левого операнда первые `n` элементов где `n` количество элементов в правом операнде, тогда как операция деления удаляет из левого операнда все элементы находящиеся в правом.

3. Исходный код

lab2.h

```
//  
// Created by user on 3/1/2024.  
//
```

```
#ifndef CMAKEPROJECT_LAB2_H  
#define CMAKEPROJECT_LAB2_H
```

```
#include "../libs/libs.h"  
#include "Point.h"  
#include "MyStack.h"
```

```
int lab2();
```

```
#endif //CMAKEPROJECT_LAB2_H
```

lab2.cpp

```
//  
// Created by user on 3/1/2024.  
//
```

```
#include "lab2.h"  
using namespace std;  
int lab2(){
```

```
    int inp;  
    double inpd;  
    do{  
        cout<<"Choose class you want to work:\n";  
        cout<<"1. Point\n";  
        cout<<"2. Stack\n";  
        cout<<"3. End"<<endl;
```

```
        cin >> inp;  
        system("cls");  
        switch(inp){  
            case 1:  
            {
```

```
                Point point;  
                do{
```

```
                    cout<<"Choose option of point you want to  
use:\n";
```

```
                    cout<<"1. print values of point\n";  
                    cout<<"2. print increment & decrement\n";  
                    cout<<"3. set X\n";  
                    cout<<"4. set Y\n";
```

```

        cout<<"5.  set Z\n";
        cout<<"6.  set increment\n";
        cout<<"7.  set decrement\n";
        cout<<"8.  point++\n";
        cout<<"9.  point--\n";
        cout<<"10. ++point\n";
        cout<<"11. --point\n";
        cout<<"12. point = !point\n";
        cout<<"13. Back"<<endl;

        cin >> inp;
        system("cls");
        switch(inp){
            case 1:
                cout<<"X:  "<<point.getX()<<"  Y:
"<<point.getY()<<" Z: "<<point.getZ()<<endl;
                break;
            case 2:
                cout<<"Increment:
"<<point.getIncrement()<<"                                Decrement:
"<<point.getDecrement()<<endl;
                break;
            case 3:
                cout<<"Enter X: ";
                cin>>inpd;
                point.setX(inpd);
                break;
            case 4:
                cout<<"Enter Y: ";
                cin>>inpd;
                point.setY(inpd);
                break;
            case 5:
                cout<<"Enter Z: ";
                cin>>inpd;
                point.setZ(inpd);
                break;
            case 6:
                cout<<"Enter Increment: ";
                cin>>inpd;
                point.setIncrement(inpd);
                break;
            case 7:
                cout<<"Enter Decrement: ";
                cin>>inpd;

```

```

        point.setDecrement(inp);
        break;
    case 8:
        point++;
        break;
    case 9:
        point--;
        break;
    case 10:
        ++point;
        break;
    case 11:
        --point;
        break;
    case 12:
        point = !point;
        break;
    case 13:
        break;
    default:
        if(inp == EOF) inp = 14;
        cout<<"There is no enough command.
If you want to go back enter 13"<<endl;
        break;
    }
}while(inp != 13);
}
break;
case 2:
{
    bool usedCopy = false;
    MyStack orig,copy;
    do {
        cout << "Choose option you want to do with
stack\n";

        cout << "1.  Choose orig\n";
        cout << "2.  Choose copy\n";
        cout << "3.    add number to current
stack\n";

        cout << "4.  enter stack\n";
        cout << "5.    pop number of current
stack\n";

        cout << "6.  print current stack\n";
        cout << "7.    get length of current
stack\n";

```

```

False)\n";
False)\n";
False)\n";
(reverse)\n";

cout << "8.  clear current stack\n";
cout << "9.  current -= another\n";
cout << "10. current /= another\n";
cout << "11. current == another (True or
False)\n";
cout << "12. current > another (True or
False)\n";
cout << "13. current < another (True or
False)\n";
cout << "14.  current  =  !current
(reverse)\n";

cout << "15. make current random\n";
cout << "16. back\n";

cin >> inp;
system("cls");
switch (inp) {
    case 1:
        usedCopy = false;
        cout << orig << endl;
        break;
    case 2:
        usedCopy = true;
        cout << cpy << endl;
        break;
    case 3:
        cout << "Enter number: ";
        cin >> inpd;
        if (usedCopy) cpy.add(inpd);
        else orig.add(inpd);
        break;
    case 4:
        cout << "Please enter num of
inputs and then enter your numbers:\n";
        if (usedCopy){
            cin >> cpy;
            cout << cpy << endl;
        }
        else {
            cin >> orig;
            cout << orig << endl;
        }
        break;
    case 5:

```

```

        if (usedCopy) cout << "Poped: " <<
cpy.pop() << endl;
        else cout << "Poped: " <<
orig.pop() << endl;
        break;
    case 6:
        if (usedCopy) cout << cpy << endl;
        else cout << orig << endl;
        break;
    case 7:
        if (usedCopy) cout << "Length: "
<< cpy.getLen() << endl;
        else cout << "Length: " << orig <<
endl;
        break;
    case 8:
        if (usedCopy) cpy.clear();
        else orig.clear();
        break;
    case 9:
        if (usedCopy){
            cout << "Current before: " <<
            cout << "Another: " <<
            cpy -= orig;
            cout << "Current now: " <<
            cpy << endl;
            orig << endl;
            cpy << endl;
            orig << endl;
        }
        else{
            cout << "Current before: " <<
            cout << "Another: " <<
            orig = orig - cpy;
            cout << "Current now: " <<
            orig << endl;
        }
        break;
    case 10:
        if (usedCopy){
            cout << "Current before: " <<
            cpy << endl;
            cout << "Another: " <<
            orig << endl;

```

```

        cpy /= orig;
        cout << "Current now:      " <<
cpy << endl;
    }
    else{
        cout << "Current before: " <<
orig << endl;
        cout << "Another:          " <<
cpy << endl;
        orig = orig / cpy;
        cout << "Current now:      " <<
orig << endl;
    }
    break;
case 11:
    if (orig == cpy){
        cout << "Current: " << cpy <<
endl;
        cout << "Another: " << orig <<
endl;
        cout << "True" << endl;
    }
    else {
        cout << "Current: " << cpy <<
endl;
        cout << "Another: " << orig <<
endl;
        cout << "False" << endl;
    }
    break;
case 12:
    if (usedCopy) {
        cout << "Current: " << cpy <<
endl;
        cout << "Another: " << orig <<
endl;
        if (cpy > orig) cout << "True"
<< endl;
        else cout << "False" << endl;
    } else {
        cout << "Current: " << cpy <<
endl;
        cout << "Another: " << orig <<
endl;

```

```

        if (orig > cpy) cout << "True"
<< endl;
        else cout << "False" << endl;
    }
    break;
case 13:
    if (usedCopy) {
        cout << "Current: " << cpy <<
endl;
        cout << "Another: " << orig <<
endl;
        if (cpy < orig) cout << "True"
<< endl;
        else cout << "False" << endl;
    } else {
        cout << "Current: " << cpy <<
endl;
        cout << "Another: " << orig <<
endl;
        if (orig < cpy) cout << "True"
<< endl;
        else cout << "False" << endl;
    }
    break;
case 14:
    if (usedCopy){
        cout << "Current before: " <<
cpy << endl;
        cpy = !cpy;
        cout << "Current now:      " <<
cpy << endl;
    }
    else {
        cout << "Current before: " <<
orig << endl;
        orig = !orig;
        cout << "Current now:      " <<
orig << endl;
    }
    break;
case 15:{
    srand(time(nullptr));
    int len = rand()%6+5;
    MyStack stack;
    while(0<len--)

```



```

        stack.add(rand()%101-50);
    if (usedCopy) {
        cpy = stack;
        cout << cpy << endl;
    }
    else {
        orig = stack;
        cout << orig << endl;
    }
}
break;
case 16:
break;
default:
    if (inp == EOF) inp = 17;
    cout << "There is no enough
command. If you want to exit enter 16\n";
    break;
}
}while(inp != 16);
}
break;
case 3:
    inp = EOF;
    break;
default:
    if(inp == EOF) inp = 4;
    cout<<"There is no enough command. If you want
to exit enter 3\n";
    break;
}
}while(inp != EOF);

return 0;
}

```

3.1 Исходный код задания №1

Point.h

```

//
// Created by user on 3/1/2024.
//

#ifndef CMAKEPROJECT_POINT_H
#define CMAKEPROJECT_POINT_H
#include <iostream>

```

```

class Point {

private:
    double _x, _y, _z;
    double _increment, _decrement;

public:
    explicit Point();
    explicit Point(double);
    explicit Point(double, double, double);
    explicit Point(double, double, double, double ,double);

    Point(const Point &p);

    ~Point();

    bool setX(double);
    bool setY(double);
    bool setZ(double);
    bool setIncrement(double);
    bool setDecrement(double);

    bool turnAll(double step = 100.0);
    void print() const;

    [[nodiscard]] double getDecrement() const;
    [[nodiscard]] double getIncrement() const;
    [[nodiscard]] double getX() const;
    [[nodiscard]] double getY() const;
    [[nodiscard]] double getZ() const;

    Point operator ! () const;
    Point &operator = (const Point&);

    Point& operator -- ();
    Point operator -- (int);

    friend Point operator ++ (Point&, int);
    friend Point& operator++( Point& );

};

#endif //CMAKEPROJECT_POINT_H

```

Point.cpp

```

//
// Created by user on 3/1/2024.
//

#include "Point.h"

Point::Point() : _x(0.), _y(0.), _z(0.),
                _increment(1.), _decrement(1.){}

Point::Point(double all) {
    this->_x = this->_y = this->_z = all;
    this->_increment = this->_decrement = 1.;
}

Point::Point(double x, double y, double z) {
    this->_x = x;
    this->_y = y;
    this->_z = z;
    this->_increment = this->_decrement = 1.;
}

Point::Point(double x, double y, double z,
              double increment, double decrement) {
    this->_x = x;
    this->_y = y;
    this->_z = z;
    this->_increment = increment;
    this->_decrement = decrement;
}

Point::Point(const Point &p) {
    this->_x = p._x;
    this->_y = p._y;
    this->_z = p._z;
    this->_increment = p._increment;
    this->_decrement = p._decrement;
}

Point::~~Point() = default;

bool Point::setX(double x) {
    this->_x = x;
    return true;
}

```

```

bool Point::setY(double y) {
    this->_y = y;
    return true;
}

bool Point::setZ(double z) {
    this->_z = z;
    return true;
}

bool Point::setIncrement(double increment) {
    this->_increment = increment;
    return true;
}

bool Point::setDecrement(double decrement) {
    this->_decrement = decrement;
    return true;
}

bool Point::turnAll(double step) {
    _x += step;
    _y += step;
    _z += step;
    return true;
}

void Point::print() const {
    std::cout<<_x<<" "<<_y<<" "<<_z<<std::endl;
}

double Point::getX() const {
    return this->_x;
}

double Point::getY() const {
    return this->_y;
}

double Point::getZ() const {
    return this->_z;
}

double Point::getIncrement() const {
    return this->_increment;
}

```

```

}

double Point::getDecrement() const {
    return this->_decrement;
}

Point Point::operator!() const {
    return Point{-_x, -_y, -_z};
}

Point &Point::operator=(const Point &p) = default;

Point &Point::operator--() {
    double max = ((_x>_y?_x:_y)>_z?(_x>_y?_x:_y):_z);
    this->_x-=max;
    this->_y-=max;
    this->_z-=max;
    return *this;
}

Point Point::operator--(int) {
    Point copy{*this};
    _x-=getDecrement();
    _y-=getDecrement();
    _z-=getDecrement();
    return copy;
}

Point operator++(Point &obj, int) {
    Point copy{obj};
    obj.setX(obj.getX()+obj.getIncrement());
    obj.setY(obj.getY()+obj.getIncrement());
    obj.setZ(obj.getZ()+obj.getIncrement());
    return copy;
}

Point &operator++(Point &obj) {
    double min = (
        (obj._x<obj._y?
            obj._x:obj._y)
        <obj._z?
            (obj._x<obj._y?
                obj._x:obj._y):
            obj._z);
    obj._x=obj._x+min;

```

```

        obj._y=obj._y+min;
        obj._z=obj._z+min;
        return obj;

}

```

3.2 Исходный код задания №2

MyStack.h

```

//
// Created by user on 3/1/2024.
//

#ifndef CMAKEPROJECT_MYSTACK_H
#define CMAKEPROJECT_MYSTACK_H
#include <iostream>

class MyStack {
private:

    class Node{
    private:
        double value;
        Node* next;
        Node* prev;
    public:
        Node();
        ~Node();
        [[nodiscard]] double getValue() const;
        Node* getNext();
        Node* getPrev();
        bool setValue(double);
        bool setNext(Node*);
        bool setPrev(Node*);
    };

    Node *head,*end;
    int len;

protected:

    [[nodiscard]] bool contain(double) const;
    bool remove(double);

public:

```

```

    explicit MyStack();
    explicit MyStack(int);
    MyStack(const MyStack&);
    ~MyStack();
    bool add(double point = 0.);
    double pop();
    [[nodiscard]] bool isEmpty() const;
    [[nodiscard]] int getLen() const;
    bool clear();

    MyStack& operator = ( const MyStack&);
    MyStack operator ! () const;
    bool operator < (const MyStack&) const;
    bool operator > (const MyStack&) const;
    friend bool operator == (const MyStack&, const MyStack&);

    MyStack& operator --(const MyStack&);
    MyStack operator - (const MyStack&) const;
    friend MyStack& operator /=(MyStack&, const MyStack&);
    friend MyStack operator / (MyStack&, const MyStack&);

    friend std::ostream& operator<<(std::ostream&, const
MyStack&);
    friend std::istream& operator>>(std::istream&, MyStack&);

};

#endif //CMAKEPROJECT_MYSTACK_H

```

MyStack.cpp

```

//
// Created by user on 3/1/2024.
//

#include "MyStack.h"

MyStack::Node::Node() {
    value = 0.;
    next = nullptr;
    prev = nullptr;
}

MyStack::Node::~~Node() = default;

double MyStack::Node::getValue() const {

```

```

        return value;
    }

    MyStack::Node *MyStack::Node::getNext() {
        return next;
    }

    MyStack::Node *MyStack::Node::getPrev() {
        return prev;
    }

    bool MyStack::Node::setValue(double point) {
        value = point;
        return true;
    }

    bool MyStack::Node::setNext(Node* node) {
        next = node;
        return true;
    }

    bool MyStack::Node::setPrev(Node* node) {
        prev = node;
        return true;
    }

    MyStack::MyStack() {
        head = nullptr;
        end = nullptr;
        len = 0;
    }

    MyStack::MyStack(int numOfNuls) {
        head = nullptr;
        end = nullptr;
        len = 0;
        while(numOfNuls-->0) add();
    }

    MyStack::MyStack(const MyStack &obj) {
        head = nullptr;
        end = nullptr;
        len = 0;
        Node *cpr = obj.head;
        while(cpr!= nullptr) {

```



```

        add(cpr->getValue());
        cpr = cpr->getNext();
    }
}

MyStack::~MyStack() {
    if(isEmpty()) return;
    Node* finder = head;
    Node* deleter;
    while(finder->getNext() != nullptr){
        deleter = finder;
        finder = finder->getNext();
        delete deleter;
    }
    delete finder;
}

bool MyStack::isEmpty() const {
    if(head == nullptr && end == nullptr) return true;
    else return false;
}

bool MyStack::add(double point) {
    Node* node = new Node();
    node->setValue(point);
    if(isEmpty()){
        head = node;
    }else end->setNext(node);

    node->setPrev(end);
    node->setNext(nullptr);
    end = node;
    len++;
    return true;
}

double MyStack::pop() {
    if(isEmpty()){
        std::cout<<"Stack is empty!\n";
        return 0.;
    }

    Node* val = end;
    double p = val->getValue();
    if(val->getPrev() == nullptr){

```

```

        delete val;
        head = end = nullptr;
        len--;
        return p;
    }

    val->getPrev()->setNext(nullptr);
    end = val->getPrev();
    delete val;
    len--;
    return p;
}

int MyStack::getLen() const {
    return len;
}

bool MyStack::contain(double a) const{
    if(isEmpty()){
        return false;
    }
    Node* finder = head;

    while(finder->getNext() != nullptr){
        if(finder->getValue() == a) return true;
        finder = finder->getNext();
    }
    if(finder->getValue() == a) return true;
    return false;
}

bool MyStack::remove(double a) {
    if(isEmpty()){
        return false;
    }
    Node* finder = head;
    if(finder->getValue() == a){
        finder->getNext()->setPrev(finder->getPrev());
        head = finder->getNext();
        delete finder;
        return true;
    }
    while(finder->getNext() != nullptr){

```

```

        if(finder->getValue() == a){
            finder->getPrev()->setNext(finder->getNext());
            finder->getNext()->setPrev(finder->getPrev());
            delete finder;
            return true;
        }
        finder = finder->getNext();
    }
    if(finder->getValue() == a){
        finder->getPrev()->setNext(finder->getNext());
        end = finder->getPrev();
        delete finder;
        return true;
    }
    return true;
}

```

```

bool MyStack::clear() {
    while(!isEmpty()) pop();
    return true;
}

```

```

MyStack &MyStack::operator=( const MyStack &obj) {
    if(&obj != this){

        clear();
        Node *cpr = obj.head;
        while(cpr!= nullptr) {
            add(cpr->getValue());
            cpr = cpr->getNext();
        }
    }
    return *this;
}

```

```

bool operator==(const MyStack &orig,const MyStack &obj) {
    MyStack a,b;
    a = orig;
    b = obj;
    if(a.isEmpty() && b.isEmpty()) return true;
    else if(a.isEmpty() || b.isEmpty()) return false;
    else{
        while(!a.isEmpty() && !b.isEmpty()){
            if(a.pop()!=b.pop()) return false;
        }
    }
}

```

```

        if(a.isEmpty() && b.isEmpty()) return true;
        else return false;
    }
}

bool MyStack::operator<(const MyStack &obj) const {
    return getLen()<obj.getLen();
}

bool MyStack::operator>(const MyStack &obj) const {
    return getLen()>obj.getLen();
}

MyStack MyStack::operator!() const {
    MyStack z, res;
    z = *this;
    while(!z.isEmpty()) res.add(z.pop());
    return res;
}

MyStack &MyStack::operator--=(const MyStack &obj) {
    if(*this>obj){
        MyStack z;
        z = obj;

        while(!z.isEmpty()){
            z.pop();
            pop();
        }
    }else{
        clear();
    }
    return *this;
}

MyStack MyStack::operator-(const MyStack &obj) const {
    MyStack z;
    z = *this;
    z-=obj;
    return z;
}

MyStack &operator/=(MyStack &orig, const MyStack &obj) {
    MyStack z;
    z = obj;

```

```

        while(!z.isEmpty()){
            double a = z.pop();
            if(obj.contains(a)) orig.remove(a);
        }
        return orig;
    }

MyStack operator/(MyStack &orig, const MyStack &obj) {
    MyStack z;
    z = orig;
    z/=obj;
    return z;
}

std::ostream &operator<<(std::ostream &os, const MyStack &obj)
{
    MyStack z;
    z = !obj;
    if(z.isEmpty()) return os<<"Stack is empty!";
    while(!z.isEmpty()) os<<z.pop()<<" ";
    return os;
}

std::istream &operator>>(std::istream &in, MyStack &obj) {
    int num;
    double var;
    in>>num;
    do {
        in >> var;
        if (in)
            obj.add(var);
    }while(--num>0);

    return in;
}

```

4. Результаты работы программы

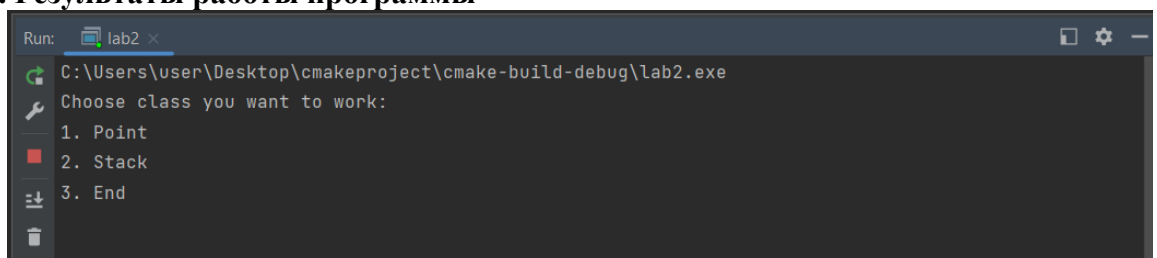
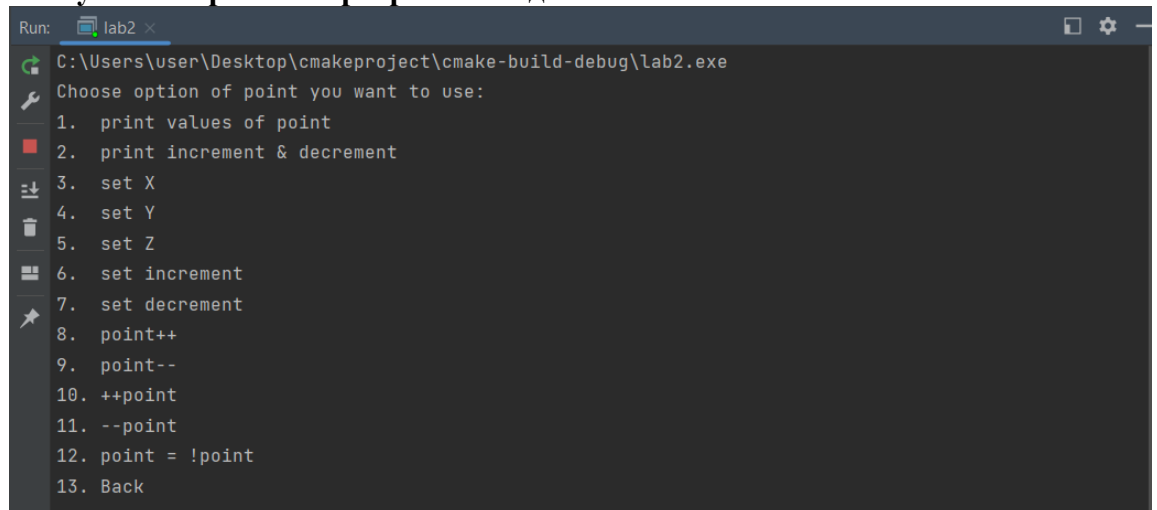


Рис. 1 – главное меню

В главном меню можно выбрать класс для работы. Для выхода из программы нужно ввести 3.

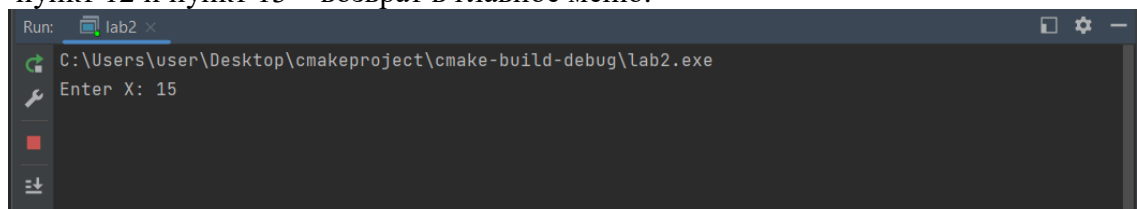
4.1 Результаты работы программы задания №1



```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
Choose option of point you want to use:
1. print values of point
2. print increment & decrement
3. set X
4. set Y
5. set Z
6. set increment
7. set decrement
8. point++
9. point--
10. ++point
11. --point
12. point = !point
13. Back
```

Рис. 2 – Меню первого класса

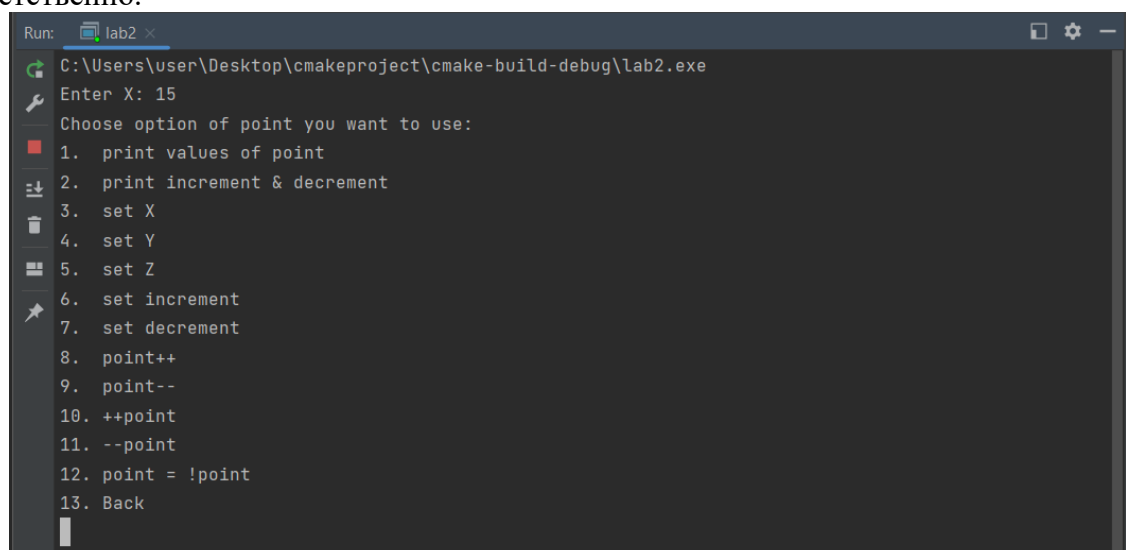
В меню представлен вывод полей класса и пользовательских значений(инкремент и декремент) – пункты 1-2, установка соответствующих значений – пункты 3-7, использование функций постфиксного и префиксного декрементов и инкрементов – пункты 8-11, реверс знака – пункт 12 и пункт 13 – возврат в главное меню.



```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
Enter X: 15
```

Рис. 3 – Ввод X

Поставим значения 15, -24, 36 в X, Y, Z и 3, 5 в инкремент и декремент соответственно.



```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
Enter X: 15
Choose option of point you want to use:
1. print values of point
2. print increment & decrement
3. set X
4. set Y
5. set Z
6. set increment
7. set decrement
8. point++
9. point--
10. ++point
11. --point
12. point = !point
13. Back
```

Рис. 4 – Результат ввода X

```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
X: 15 Y: -24 Z: 36
Choose option of point you want to use:
1. print values of point
2. print increment & decrement
3. set X
4. set Y
5. set Z
6. set increment
7. set decrement
8. point++
9. point--
10. ++point
11. --point
12. point = !point
13. Back
```

Рис. 5 – Введенные значения X, Y, Z

```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
Increment: 3 Decrement: 5
Choose option of point you want to use:
1. print values of point
2. print increment & decrement
3. set X
4. set Y
5. set Z
6. set increment
7. set decrement
8. point++
9. point--
10. ++point
11. --point
12. point = !point
13. Back
```

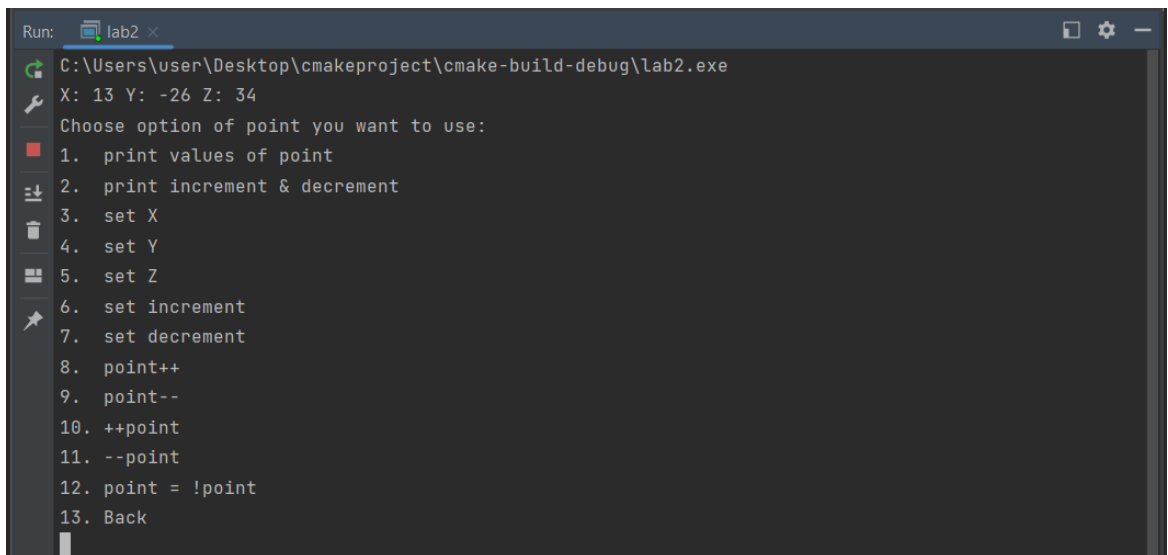
Рис. 6 – Введенные значения инкремента и декремента

Используем постфиксный инкремент. В результате значения должны увеличиться на 3.

```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
X: 18 Y: -21 Z: 39
Choose option of point you want to use:
1. print values of point
2. print increment & decrement
3. set X
4. set Y
5. set Z
6. set increment
7. set decrement
8. point++
9. point--
10. ++point
11. --point
12. point = !point
13. Back
```

Рис. 7 – Результат постфиксного инкремента

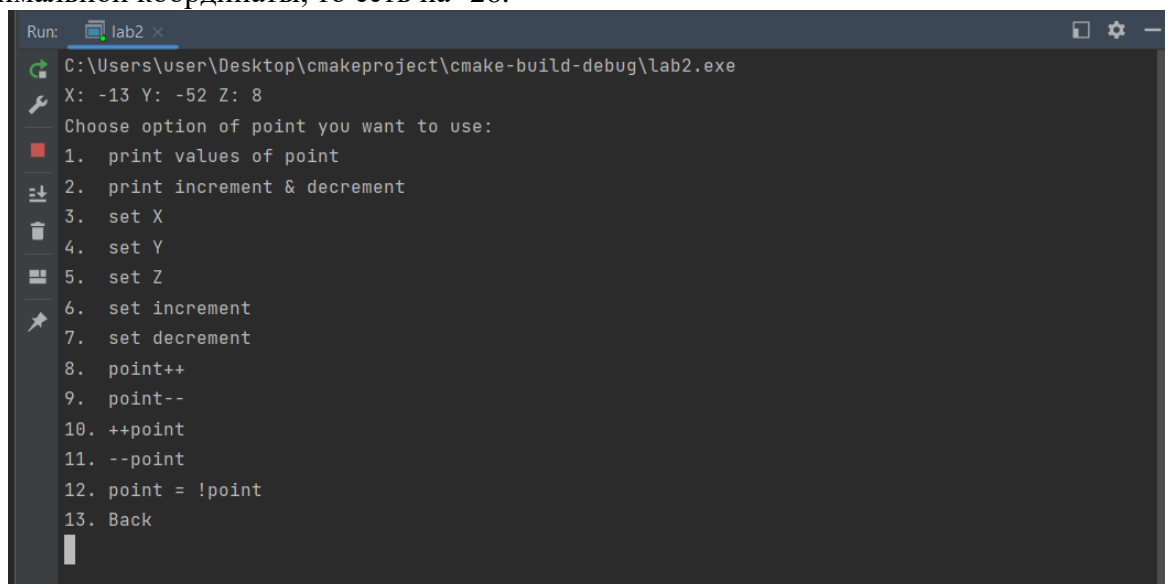
Используем постфиксный декремент. В результате значения должны уменьшиться на 5.



```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
X: 13 Y: -26 Z: 34
Choose option of point you want to use:
1. print values of point
2. print increment & decrement
3. set X
4. set Y
5. set Z
6. set increment
7. set decrement
8. point++
9. point--
10. ++point
11. --point
12. point = !point
13. Back
```

Рис. 8 – Результат постфиксного декремента

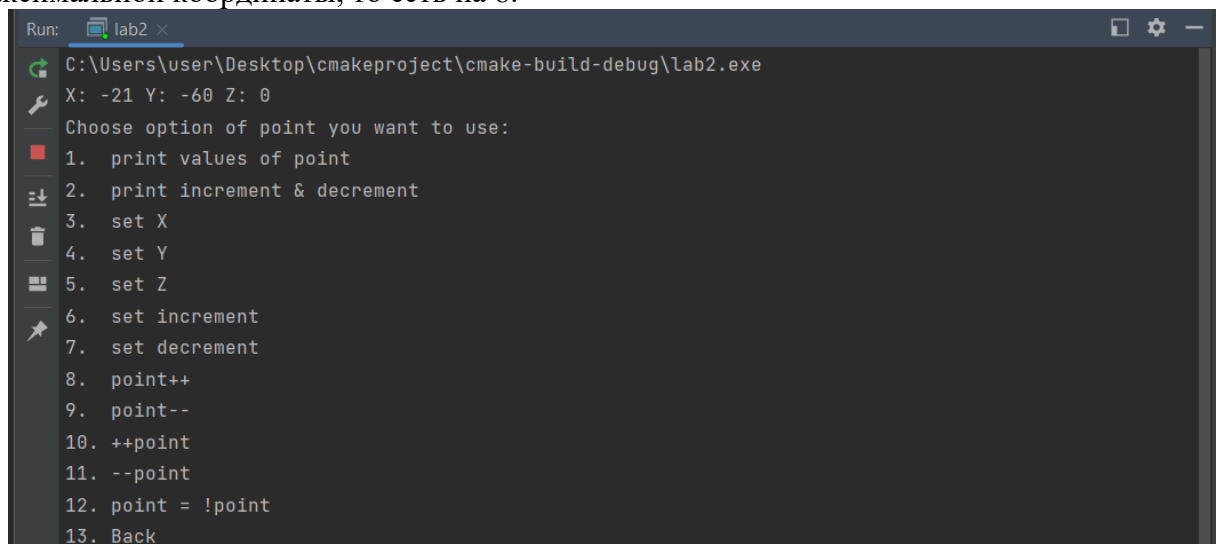
Используем префиксный инкремент. В результате значения должны увеличиться на значение минимальной координаты, то есть на -26.



```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
X: -13 Y: -52 Z: 8
Choose option of point you want to use:
1. print values of point
2. print increment & decrement
3. set X
4. set Y
5. set Z
6. set increment
7. set decrement
8. point++
9. point--
10. ++point
11. --point
12. point = !point
13. Back
```

Рис. 9 – Результат префиксного инкремента

Используем префиксный декремент. В результате значения должны уменьшиться на значение максимальной координаты, то есть на 8.



```
Run: lab2 x
C:\Users\user\Desktop\cmakeproject\cmake-build-debug\lab2.exe
X: -21 Y: -60 Z: 0
Choose option of point you want to use:
1. print values of point
2. print increment & decrement
3. set X
4. set Y
5. set Z
6. set increment
7. set decrement
8. point++
9. point--
10. ++point
11. --point
12. point = !point
13. Back
```

Рис. 10 – Результат префиксного декремента

Поставим значение $Z = 1$ и используем смену знака.

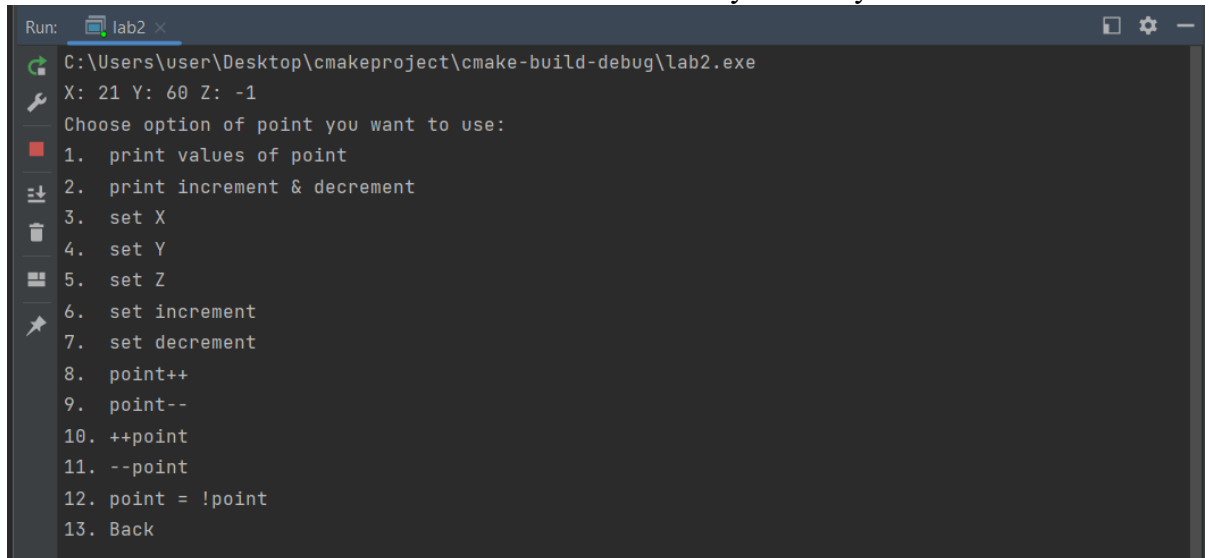


Рис. 11 – Результат смены знака

4.2 Результаты работы программы задания №2

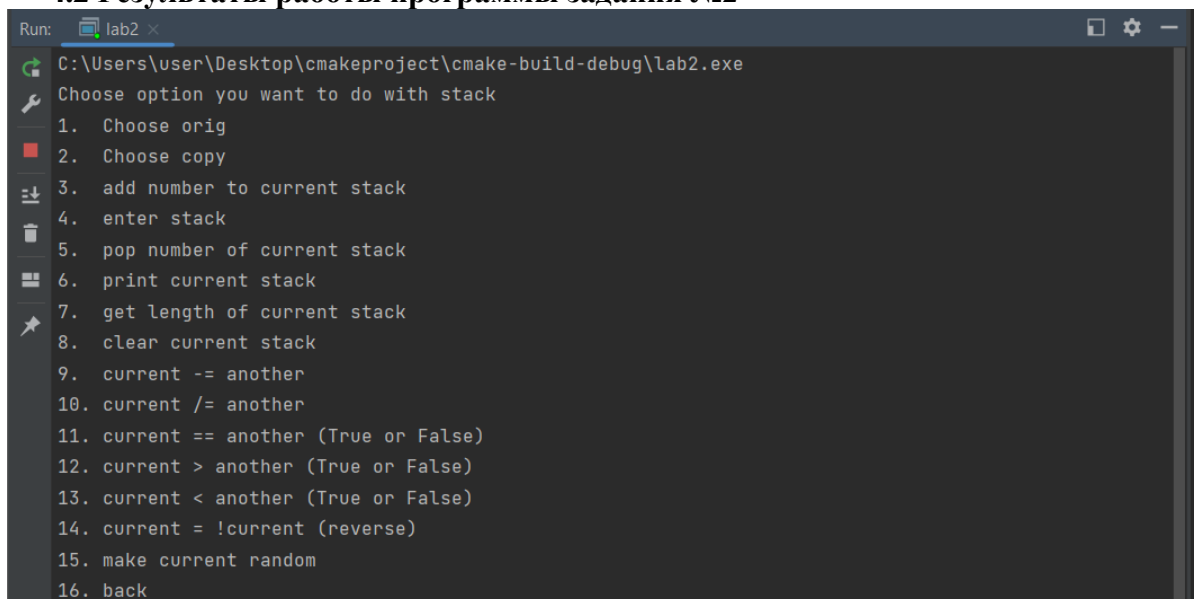


Рис. 12 – Меню работы со стеком

Для работы предоставлено два стека, оригинал и копия. Команды 1-2 отвечают за выбор стека, команда 3 запрашивает 1 число на ввод. Команда 4 запрашивает n и n чисел после него на ввод. Команда 5 удаляет одно число из стека. Команда 6 выводит стек на экран. Команда 7 выводит длину стека, 8я очищает его. Команды 9 и 10 проводят операции со стеками, команды 11-13 сравнивают стеки, 14 проводит реверс и 15 очищает и заполняет случайными значениями.

Заполним случайными значениями об стека.

```
Run: lab2 x
-16 16 17 -48 -19 -1 -25 24 38
Choose option you want to do with stack
1. Choose orig
2. Choose copy
3. add number to current stack
4. enter stack
5. pop number of current stack
6. print current stack
7. get length of current stack
8. clear current stack
9. current -= another
10. current /= another
11. current == another (True or False)
12. current > another (True or False)
13. current < another (True or False)
14. current = !current (reverse)
15. make current random
```

Рис. 13 – orig

```
Run: lab2 x
-41 7 35 -28 33 35 -26 41 5
Choose option you want to do with stack
1. Choose orig
2. Choose copy
3. add number to current stack
4. enter stack
5. pop number of current stack
6. print current stack
7. get length of current stack
8. clear current stack
9. current -= another
10. current /= another
11. current == another (True or False)
12. current > another (True or False)
13. current < another (True or False)
14. current = !current (reverse)
15. make current random
16. back
```

Рис. 14 – copy

Разделим orig на copy. Так как одинаковых членов нет, orig не изменится.

```
Run: lab2 x
Current before: -16 16 17 -48 -19 -1 -25 24 38
Another:        -41 7 35 -28 33 35 -26 41 5
Current now:    -16 16 17 -48 -19 -1 -25 24 38
Choose option you want to do with stack
1. Choose orig
2. Choose copy
3. add number to current stack
4. enter stack
5. pop number of current stack
6. print current stack
7. get length of current stack
8. clear current stack
9. current -= another
10. current /= another
11. current == another (True or False)
12. current > another (True or False)
13. current < another (True or False)
14. current = !current (reverse)
15. make current random
16. back
```

Рис. 15 – Результат деления

Реверсируем orig.

```
Run: lab2 x
Current before: -16 16 17 -48 -19 -1 -25 24 38
Current now:    38 24 -25 -1 -19 -48 17 16 -16
Choose option you want to do with stack
1. Choose orig
2. Choose copy
3. add number to current stack
4. enter stack
5. pop number of current stack
6. print current stack
7. get length of current stack
8. clear current stack
9. current -= another
10. current /= another
11. current == another (True or False)
12. current > another (True or False)
13. current < another (True or False)
14. current = !current (reverse)
15. make current random
16. back
```

Рис. 16 – Результат реверсирования

Сравним оригинал и копию. Оригинал не больше копии, а значит 12й пункт должен дать True.

```
False
Choose option you want to do with stack
1. Choose orig
2. Choose copy
3. add number to current stack
4. enter stack
5. pop number of current stack
6. print current stack
7. get length of current stack
8. clear current stack
9. current -= another
10. current /= another
11. current == another (True or False)
12. current > another (True or False)
13. current < another (True or False)
14. current = !current (reverse)
15. make current random
16. back
```

Рис. 17 – Результат сравнения

5. Выводы

В процессе работы я научился работать с перегрузкой операторов на языке программирования C++. Объектно реализовал классы стэка и очереди, и перегрузил унарные и бинарные операторы у обоих.

Результатом работы стала программа для работы классом координат и с двумя стэками – оригиналом и копией. Программа работает успешно, так как ожидаемые результаты работы совпадают с фактическими.