МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет «МИФИ»

**Институт интеллектуальных кибернетических систем**

**кафедра “Кибернетика”(22)**

“Объектно-Ориентированное Программирование”

**Лабораторная работа по ООП:**

«Лабораторная работа № 2. Изучение конструкций ООП в C++»

*Выполнил: Локтионов А.А. гр. М19-514*

*Преподаватель: Шапкин П.А.*

Москва

2020 г.

**Задание 1.** Провести исследование порядка вызова конструкторов/деструкторов при наследовании: составить программу, отображающую порядок выполнения операций при создании и удалении объекта.

#include <iostream>

using namespace std;

class Father {

public:

Father() {

cout << "Here Father constructor is called" << endl;

}

~Father() {

cout << "Here Father destructor is called" << endl;

}

};

class Son : public Father

{

public:

Son() {

cout << "Here Son constructor is called" << endl;

};

~Son() {

cout << "Here Son destructor is called" << endl;

};

};

int main1() {

auto father = new Father();

delete father;

cout << endl;

auto son = new Son();

delete son;

cout << endl;

Father\* fosterSon = new Son();

delete fosterSon;

return 0;

}

**Задание 2.** Разработать систему классов для представления структуры синтаксического дерева арифметических выражений и вычисления его значения

#include <iostream>

using namespace std;

class Expr

{

public:

virtual int eval() = 0;

};

class Num : public Expr

{

private:

int number;

public:

Num(int n) : number(n) {};

int eval() override { return number; };

};

class Add : public Expr

{

private:

Expr\* l;

Expr\* r;

public:

Add(Expr\* left, Expr\* right) : l(left), r(right) {};

int eval() override { return l->eval() + r->eval(); };

};

class Subtract : public Expr

{

private:

Expr\* l;

Expr\* r;

public:

Subtract(Expr\* left, Expr\* right) : l(left), r(right) {};

int eval() override { return l->eval() - r->eval(); };

};

int main2() {

auto tree = new Add(new Add(new Num(4), new Num(5)), new Subtract(new Num(9), new Num(1))); //=(4+5)+(9-1)

cout << tree->eval() << endl;

delete tree;

return 0;

}

Задание 3. Разработать систему классов, реализующих структуры структуры данных:

• Базовый класс:

Collection(int n) — коллекция чисел, хранящая элементы в массиве длины n: – метод int get(int i) возвращает i-й элемент.

• Производные классы:

1. Stack — стэк: – метод void put(int element) помещает элемент в конец стэка; – метод int take() выталкивает из стэка последний элемент.

2. Queue — очередь: – метод void put(int element) помещает элемент в начало очереди; – метод int take() извлекает последний элемент очереди. Написать определения классов на языке C++.

Написать следующую программу:

1. Создать массив коллекций.

2. Заполнить массив коллекциями различных типов (стэками и очередями).

3. В цикле: используя методы put и take провести запись и чтение значений.

4. В цикле: распечатать элементы коллекций массива.

Проверить программу на предмет отсутствия утечек памяти.

#include <iostream>

#include <iomanip>

using namespace std;

class Collection

{

protected:

int\* array;

int size;

public:

Collection(int n) : size(n) { array = new int[n]; memset(array, 0, n \* sizeof(int)); }

~Collection() { delete[] array; }

int get(int i) { return (i>=0&&i<size) ? i[array] : -1 ; }

virtual void put(int element) = 0;

virtual int take() = 0;

};

class Stack : public Collection

{

private:

int pos;

public:

explicit Stack(int n) : Collection(n) { pos = 0; }

void put(int element) override { (pos < size) ? array[pos++] = element : true; }

int take() override { return ((pos > 0) ? array[--pos] : 0 ); }

};

class Queue : public Collection

{

private:

int pos;

public:

explicit Queue(int n) : Collection(n) { pos = size; }

void put(int element) override { (pos > 0) ? array[--pos] = element : true; }

int take() override { return ((pos < size) ? array[pos++] : 0); }

};

int main() {

Collection\*\* array = new Collection\*[10];

for (int i = 0; i < 10; i++)

{

if (((bool)((i % 3) % 2))) {

array[i] = new Queue(i \* 3);

}

else {

array[i] = new Stack(i \* 2);

}

}

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 100; j++)

{

array[i]->put(j% (i+1) + i % (j+1) + 1);

}

}

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 3; j++)

{

array[i]->take();

}

}

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 30; j++)

{

cout << setw(3) << array[i]->get(j) << ' ';

}

cout << endl;

}

return 0;

}