Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

**Кафедра автоматизированных систем управления (АСУ)**

**НАЗВАНИЕ РАБОТЫ**

**Отчет по лабораторной работе № 2 по дисциплине**

**«Объектно ориентированное программирование»**

Выполнил: Полтавский Н.С.

«20» ноября 2019 г.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Томск — 2019

СОДЕРЖАНИЕ

1.Содержание

2. Тема работы 2

3. Цель работы 3

4. Интструмены для выполнения 3

5. Алгоритм решения задачи 3-6

7. Результаты работы программы 6

8. Выводы 6

9. Приложение А. Листинг программы 7-8

**Тема работы** - Динамические массивы объектов, деструкторы объектов. Полиморфизм.

**Цель работы –** На практике проверить навыки по созданию динамического массива объектов. Деструктора объекта. Проработать два типа полиморфизма: принудительное приведение типа, перегрузка функций и перегрузка операторов

(унарных и бинарных).

**Интструмены для выполнения:** данная программа для решения задания лабораторной работы выполнялась на языке С++ в среде разработке VS 2019 на ОС Windows 10 1909.

**Алгоритм решения задачи**

* Подключаем необходимые библиотеки

- <iostream>

- <math.h>

- <cmath>

- <iomanip>

- <tchar.h>

- <conio.h>

- <iostream>

- <Windows.h>

- <mmsystem.h>

- <fstream>

- <string>

- <iostream>

- <stdio.h>

- <sys/types.h>

- <process.h>

- <thread>

- "office.h"

- "printer.h"

- pragma once

* Файлы из прошлого проекта - main.cpp и main.h переименовываем в printer.cpp и printer.h
* Добавляем необходимые заголовки в файлы
* Настраиваем используемое именное пространство
* Определим объект Office в файле и, который будет содержать динамический массив объектов Printer. Создайте два файла office.h office.cpp и включите их в проект.
* Задаем ему поля данных(поле название из массива char я самостоятельно заменил на string по сути своей массив char с добавленными в класс функциями работы) :
  + Int размер массива
  + Указатель на массив (Printer \*)
* Закрываем данные класса в доступ Private
* Пишем базовую функцию инициализации класса Принтер для инициализации класса с параметрами размера и инициализацией массива.
* Пишем деструктор объекта Office
* Описываем функцию get\_office вывода данных из переменных класса Office в стандартный поток вывода консоли используя функцию get\_prn класса Printer и цикл for и перегрузку к ней для возвращения объекта Printer&.
* Описываем функцию set\_office ввода данных в переменные класса Office по средствам передачи в фукцию ссылки на Printer& и указателя на место в массиве класса Office
* Пишем функцию get\_office\_size возвращающую int кфяьук класса Office
* Перегружаем функцию гет\_офис для возвращения класса Printer Printer& get\_office по введенной позиции указателя инт.
* Создаем и заполняем функцию main.
* Формируем правильное отображение кирилицы
* Создаем множество строк models для хранения имен моделей принтеров
* Затем создаем объект с именем Second типа Office размером пять записей, т.е. Office Second(5).
* Ниже с помощью цикла for заполняем массив данными из сета models.
* В теле цикла создаем объект Printer, проинициализировав все его данные, и с помощью функции PutPrinter занесите

объект в массив.

* Выводим массив на экран с помощью строки кода:

Office.get\_office();. Получилось. УРА.

* Пишем функцию приведения типа. Для этого с помощью ключевого слова operator описываем функцию объекта Printer, которая преобразует тип Printer в double. Пусть функция возвращает сумму листов принтера

Printer::operator double() {

return this->summ;

}

* Проверяем функцию преобразования типа. В функции

int main() далее определим переменную double sum и присвоем ей

объект Printer, например:

double ыгь = Second.get\_office(3);

То есть совершается неявное преобразование из типа Printer

в тип double при обращении к объекту. Для проверки выводим значение переменной на экран.

* Определим в объекте Second Office две функции с одинаковым именем,

например double summ(); и double summ(int limit);. Делаем их по аналогии с прошлой функцией приведения. Используя выдачу значения Printer из цикла for.

* Проверяем работу функций с помощью вывода значений на стдаут.
* Перегружаем оператор индексирования. Если раньше, чтобы обратиться к элементу массива, нам необходимо было вызывать функцию Second.get\_printer(int), то, определив оператор индексирования, мы будем использовать только квадратные скобки. Сравните

две строки кода:

Printer man = Second.get\_printer (2);

Printer man = Second [2];

В объекте Office с помощью ключевого слова operator определяем оператор индексирования, например:

Printer& operator[](int i).

В теле оператора пишем ретурн на объект принтер возвращающий i-ый элемент массива, т.е. объект Printer.

* Выведим на экран с помощью оператора индексирования любой один элемент массива Office, например первый с перегрузкой и 4 без.
* Перегрузим бинарный оператор, например оператор
* сложения (+), для объекта Printer. Пусть оператор сложения будет возвращать суммарное количество печати двух принтеров. Опишем в объекте Printer данный оператор как дружественную функцию с помощью ключевого слова friend.
* В файле Printer.cpp определите оператор сложения, например:

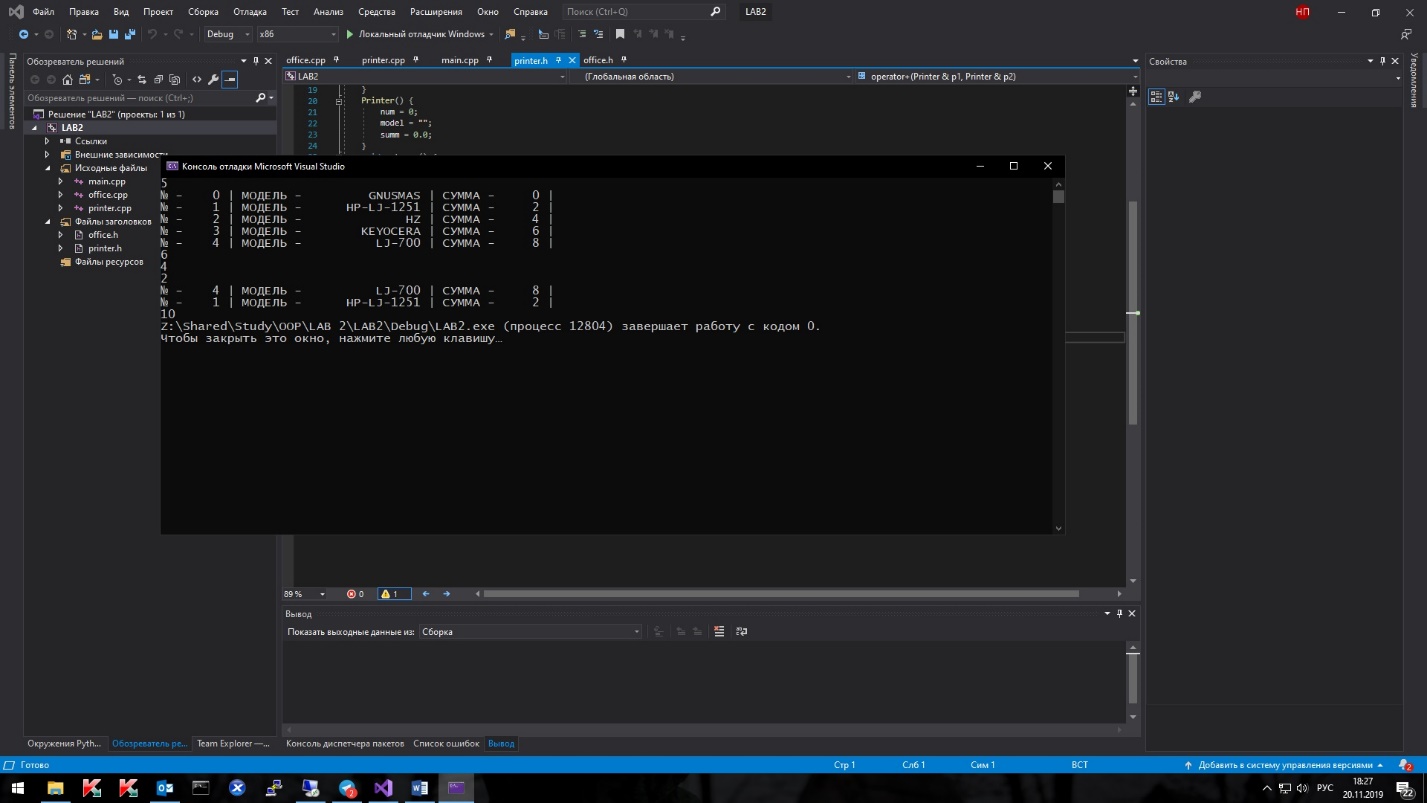
double operator + (Printer& p1, Printer& p2)

{ return (p1.summ + p2.summ); }

Здесь мы напрямую обращаемся к закрытому полю Summ объекта Printer.

* Проверим работу оператора с помощью вызова функции с аргументами в стдаут

**Результат работы программы**

****

**Выводы**

При выполнении данной лабараторной работы я повторил и освоил создание динамического массива объектов, деструкторов. Реализовал два типа полиморфизма: принудительное приведение типа, перегрузка функций и перегрузка операторов.

**Приложение А**

**Листинг программы**

**MAIN.CPP**

#include <iostream>

#include <math.h>

#include <cmath>

#include <iomanip>

#include <tchar.h>

#include <conio.h>

#include <iostream>

#include <Windows.h>

#include <fstream>

#include <string>

#include <iostream>

#include <stdio.h>

#include <vector>

#include <set>

#include <algorithm>

#include "office.h"

#include "printer.h"

using namespace std;

#pragma once

int main() {

setlocale(LC\_ALL, "Russian");

int k = 0;

set <string> models = { "LJ-700" , "KEYOCERA", "HP-LJ-1251", "GNUSMAS", "PHILIPS", "HZ" };

Office Second(6);

for (auto now = models.begin(); now != models.end(); now++) {

Printer A(k, \*now, (float)k + k);

Second.set\_office(k, A);

k++;

}

cout <<Second.get\_office\_size() << endl;

Second.get\_office();

double sum = Second.get\_office(3);

cout << sum << endl;

cout << Second.summ() << endl;

cout << Second.summ(4) << endl;

Printer B = Second.get\_office(4);

Printer C = Second[1];

B.get\_prn();

C.get\_prn();

cout << Second[2] + Second[3];

return 0;

}

**PRINTER.H**

#include <string>

#include <iomanip>

#include <iostream>

using namespace std;

#pragma once

class Printer {

private:

string model;

int num;

float summ;

public:

Printer(int a, string b, float d) {

num = a;

model = b;

summ = d;

}

Printer() {

num = 0;

model = "";

summ = 0.0;

}

void get\_prn() {

cout << "№ - " << setw(4) << num << " | " << "МОДЕЛЬ - " << setw(15) << model << " | " << "СУММА - " << setw(4) << summ << " | " << endl;

}

void set\_prn(int a, string b, float d) {

num = a;

model = b;

summ = d;

}

void set\_prn() {

cin >> num;

cin >> model;

cin >> summ;

}

friend double operator+(Printer& p1, Printer& p2) {

return p1.summ + p2.summ;

}

operator double() {

return this->summ;

}

};

**OFFICE.H**

#include <string>

#include <iomanip>

using namespace std;

#include "printer.h"

#include <iostream>

#pragma once

class Office {

private:

int size;

Printer \*prn;

public:

Office(int S) {

size = S;

prn = new Printer[this->size];

}

~Office() {

delete[] prn;

}

void get\_office() {

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

prn[i].get\_prn();

}

}

int get\_office\_size() {

return size;

}

void set\_office(int i, Printer& p) {

this->size = i;

this->prn[i] = p;

}

Printer& get\_office(int i) {

return prn[i];

}

double summ() {

double s = 0;

double res = 0;

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

s += prn[i].operator double();

}

res = s / size;

return res;

}

double summ(int limit) {

int ptr = 0;

double s = 0;

double res = 0;

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

if (limit >= prn[i].operator double()) {

s += prn[i].operator double();

ptr++;

}

}

res = s / ptr;

return res;

}

Printer& operator[](int i){

return prn[i];

}

};

**PRINTER.CPP**

#include <cmath>

#include <iomanip>

#include <tchar.h>

#include <conio.h>

#include <iostream>

#include <Windows.h>

#include <fstream>

#include <string>

#include <iostream>

#include <stdio.h>

#include "printer.h"

#include "office.h"

using namespace std;

#pragma once

Printer print(0, "HP LJ 335", 100);

**OFFICE.CPP**

#include <iostream>

#include <string>

#include "office.h"

#include "printer.h"

using namespace std;

#pragma once

Office First(6);

Printer \*prn = new Printer[6];