Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный университет”

Кафедра ИИТ

Лабораторная работа №3

По дисциплине “Языки программирования”

Вариант №8

Выполнил:

Крук Д.Н. (ПО-7,2)

Проверил:

Дряпко. А. В.

Дата выполнения:

28.10.21

Брест 2021

**Лабораторная работа №3**

**“ Перегрузка операций. Исключения”**

**Вариант 8**

**Общие требования:** в начале программы вывести задание; в процессе работы выводить подсказки пользователю (что ему нужно ввести, чтобы продолжить выполнение программы). Иерархию классов следует взять из лабораторной работы №3. После работы программы вся динамически выделенная память должна быть освобождена. Класс коллекция должна иметь методы для изменения количества хранимых объектов: добавление в конец, вставка, усечение, удаление из середины. При обращении к элементам с несуществующим индексом или при некорректном изменении количества должно выбрасываться исключение.

Взаимодействие с пользователем организовать в виде простого меню, обеспечивающего возможность переопределения исходных данных и завершение работы программы.

**Задание:** написать программу, в которой описана иерархия классов: геометрическая фигура **(эллипс, квадрат, треугольник)**. Описать класс для хранения коллекций фигур (массива указателей на базовый класс), в котором перегрузить операцию “[ ]”, а также реализовать функции подсчёта общей площади и периметра. Для базового класса и его потомков перегрузить операции “==”, “! =”, “=”. Продемонстрировать работу операторов.

**Код программы:**

**Figure.h:**

#ifndef INC\_3\_FIGURE\_H  
#define INC\_3\_FIGURE\_H  
  
#include <iostream>  
#include "math.h"  
  
using namespace std;  
  
  
class Figure {  
public:  
 virtual double square() = 0;  
 virtual double perimeter() = 0;  
 virtual void print() = 0;  
};  
  
  
#endif

**Ellipses.h:**

#ifndef INC\_3\_ELLIPSES\_H  
#define INC\_3\_ELLIPSES\_H  
  
#include "Figure.h"  
  
class Ellipses : public Figure {  
 double a;  
 double b;  
public:  
 Ellipses();  
 Ellipses(double, double);  
 double square() override;  
 double perimeter() override;  
 void print() override;  
 bool operator==(const Ellipses&);  
 bool operator!=(const Ellipses&);  
 Ellipses& operator=(const Ellipses&);  
 ~Ellipses();  
};  
  
  
#endif

**Ellipses.cpp:**

#include "Ellipses.h"  
  
Ellipses::Ellipses() {  
 cout << "Введите малую ось: ";  
 cin >> a;  
 cout << "Введите большую ось: ";  
 cin >> b;  
 cout << endl;  
}  
  
Ellipses::Ellipses(double a, double b) {  
 this->a = a;  
 this->b = b;  
}  
  
double Ellipses::square() {  
 return a \* b \* M\_PI;  
}  
  
double Ellipses::perimeter() {  
 return (a + b) \* M\_PI;  
}  
  
void Ellipses::print() {  
 cout << "Элипс:" << endl  
 << "Малая ось: " << a << endl  
 << "Большая ось: " << b << endl;  
}  
  
bool Ellipses::operator==(const Ellipses &ellipse) {  
 return (a == ellipse.a && b == ellipse.b);  
}  
  
bool Ellipses::operator!=(const Ellipses &ellipse) {  
 return (a != ellipse.a || b != ellipse.b);  
}  
  
Ellipses &Ellipses::operator=(const Ellipses &ellipse) {  
 a = ellipse.a;  
 b = ellipse.b;  
 return \*this;  
}  
  
Ellipses::~Ellipses() {  
 a = 0;  
 b = 0;  
}

**Foursquare.h:**

#ifndef INC\_3\_FOURSQUARE\_H  
#define INC\_3\_FOURSQUARE\_H  
  
#include "Figure.h"  
  
class Foursquare : public Figure {  
 double a;  
public:  
 Foursquare();  
 Foursquare(double);  
 double square() override;  
 double perimeter() override;  
 void print() override;  
 bool operator==(const Foursquare &);  
 bool operator!=(const Foursquare &);  
 Foursquare &operator=(const Foursquare &);  
 ~Foursquare();  
};  
  
  
#endif //INC\_3\_FOURSQUARE\_H

**Foursquare.cpp:**

#include "Foursquare.h"  
  
Foursquare::Foursquare() {  
 cout << "Введите сторону: ";  
 cin >> a;  
}  
  
Foursquare::Foursquare(double a) {  
 this->a = a;  
}  
  
double Foursquare::square() {  
 return a \* a;  
}  
  
double Foursquare::perimeter() {  
 return 4 \* a;  
}  
  
void Foursquare::print() {  
 cout << "Квадрат: " << endl  
 << "Сторона: " << a << endl;  
}  
  
bool Foursquare::operator==(const Foursquare &foursquare) {  
 return (a == foursquare.a);  
}  
  
bool Foursquare::operator!=(const Foursquare &foursquare) {  
 return (a != foursquare.a);  
}  
  
Foursquare &Foursquare::operator=(const Foursquare &foursquare) {  
 a = foursquare.a;  
 return \*this;  
}  
  
Foursquare::~Foursquare() {  
 a = 0;  
}

**Trapezoid.h:**

#ifndef INC\_3\_TRAPEZOID\_H  
#define INC\_3\_TRAPEZOID\_H  
  
#include "Figure.h"  
  
class Trapezoid : public Figure{  
 double bot\_base;  
 double top\_base;  
 double h;  
public:  
 Trapezoid();  
 Trapezoid(double, double, double);  
 double square() override;  
 double perimeter() override;  
 void print() override;  
 bool operator==(const Trapezoid &);  
 bool operator!=(const Trapezoid &);  
 Trapezoid &operator=(const Trapezoid &);  
 ~Trapezoid();  
};  
  
  
#endif

**Trapezoid.cpp:**

#include "Trapezoid.h"  
  
Trapezoid::Trapezoid() {  
 cout << "Введите верхнее основание: ";  
 cin >> bot\_base;  
 cout << "Введите нижнее основание: ";  
 cin >> top\_base;  
 cout << "Введите высоту: ";  
 cin >> h;  
 cout << endl;  
}  
  
Trapezoid::Trapezoid(double bot\_base, double top\_base, double h) {  
 this->bot\_base = bot\_base;  
 this->top\_base = top\_base;  
 this->h = h;  
}  
  
double Trapezoid::square() {  
 return (bot\_base + top\_base) / 2 \* h;  
}  
  
double Trapezoid::perimeter() {  
 double side = pow((bot\_base - top\_base) / 2, 2) + pow(h, 2);  
 return bot\_base + top\_base + 2 \* (sqrt(side));  
}  
  
void Trapezoid::print() {  
 cout << "Трапеция: " << endl  
 << "Верхнее основание: " << top\_base << endl  
 << "Нижнее основание: " << bot\_base << endl  
 << "Высота: " << h << endl;  
}  
  
bool Trapezoid::operator==(const Trapezoid &trapezoid) {  
 return bot\_base == trapezoid.bot\_base && top\_base == trapezoid.top\_base && h == trapezoid.h;  
}  
  
bool Trapezoid::operator!=(const Trapezoid &trapezoid) {  
 return bot\_base != trapezoid.bot\_base || top\_base != trapezoid.top\_base || h != trapezoid.h;  
}  
  
Trapezoid &Trapezoid::operator=(const Trapezoid &trapezoid) {  
 bot\_base = trapezoid.bot\_base;  
 top\_base = trapezoid.top\_base;  
 h = trapezoid.h;  
 return \*this;  
}  
  
Trapezoid::~Trapezoid() {  
 bot\_base = 0;  
 top\_base = 0;  
 h = 0;  
}

**LFigure.h:**

#ifndef INC\_3\_LFIGURE\_H  
#define INC\_3\_LFIGURE\_H  
  
#include "limits"  
#include "Figure.h"  
#include "Ellipses.h"  
#include "Foursquare.h"  
#include "Trapezoid.h"  
#include "windows.h"  
  
typedef Figure \*pFigure;  
  
class LFigure {  
 pFigure \*m\_figure;  
 int size;  
public:  
 LFigure(int);  
 Figure\*& operator[](int);  
 void ins(Figure \*&);  
 void add(Figure \*&);  
 void val();  
 void sum();  
 void truncation();  
 void del\_middle();  
 void show();  
 ~LFigure();  
};  
  
  
#endif

**LFigure.cpp:**

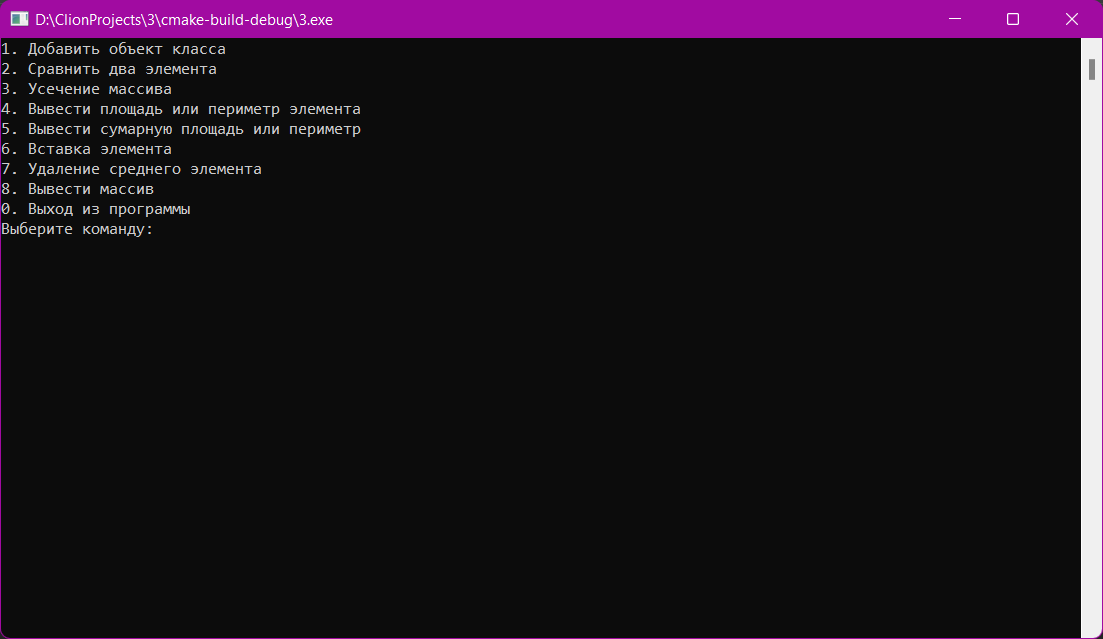
#include "LFigure.h"  
  
LFigure::LFigure(int size) {  
 this->size = size - 1;  
 m\_figure = new pFigure[size];  
 for (int i = 0; i < size; i++)  
 m\_figure[i] = nullptr;  
}  
  
Figure \*&LFigure::operator[](int n) {  
 if (n < 0 && n > size) throw LFigure(size);  
 return m\_figure[n];  
}  
  
void LFigure::ins(Figure \*&figure) {  
 int ind;  
 cout << "Введите номер вставки: ";  
 cin >> ind;  
  
 if (ind < 0 || ind > size) throw LFigure(size);  
 if (m\_figure[ind] && m\_figure[size]) throw;  
 if (m\_figure[ind])  
 for (int i = size - 1; i > ind; i--)  
 m\_figure[i] = m\_figure[i - 1];  
 m\_figure[ind] = figure;  
}  
  
void LFigure::add(Figure \*&figure) {  
 if (m\_figure[size]) throw LFigure(size);  
 int i;  
 for (i = size; i >= 0 && !m\_figure[i]; i--);  
 m\_figure[i + 1] = figure;  
}  
  
void LFigure::truncation() {  
 int ind;  
 cout << "Индекс усечения: ";  
 cin >> ind;  
  
 if (ind < 0 && ind > size) throw LFigure(size);  
 for (int i = ind; i < size; i++)  
 m\_figure[i] = nullptr;  
}  
  
void LFigure::val() {  
 cout << "1. Площадь" << endl  
 << "2. Периметр" << endl  
 << "Выберите команду: ";  
 int c, ind;  
 cin >> c;  
 cout << "Введите индекс элемента: ";  
 cin >> ind;  
 if (ind >= size) throw LFigure(size);  
 if (m\_figure[ind] == nullptr) throw ind;  
 if (c == 1) {  
 double s = m\_figure[ind]->square();  
 if (s > numeric\_limits<double>::max()) throw 0.1;  
 cout << "Площадь: " << s << endl;  
 }  
 if (c == 2) {  
 double p = m\_figure[ind]->perimeter();  
 if (p > numeric\_limits<double>::max()) throw 0.1;  
 cout << "Периметр: " << p << endl;  
 }  
}  
  
void LFigure::sum() {  
 double sum = 0;  
 int c;  
 cout << "1. Площадь" << endl  
 << "2. Периметр" << endl  
 << "Выберите команду: ";  
 cin >> c;  
 double (Figure::\*f)();  
  
 if (c == 1)  
 f = &Figure::square;  
 if (c == 2)  
 f = &Figure::perimeter;  
  
 for (int i = 0; i <= size; i++)  
 if (m\_figure[i] != nullptr)  
 sum += (m\_figure[i]->\*f)();  
  
 if (sum > numeric\_limits<double>::max()) throw 0.1;  
  
 if (c == 1)  
 cout << "Сумарная площадь: " << sum << endl;  
 else if (c == 2)  
 cout << "Сумарный периметр: " << sum << endl;  
}  
  
void LFigure::del\_middle() {  
 for (int i = size / 2; i < size; i++)  
 m\_figure[i] = m\_figure[i + 1];  
 m\_figure[size] = nullptr;  
}  
  
void LFigure::show() {  
 for (int i = 0; i <= size; i++)  
 if (m\_figure[i] != nullptr) {  
 cout << i << endl;  
 m\_figure[i]->print();  
 cout << endl;  
 }  
}  
  
LFigure::~LFigure() {  
 for (int i = 0; i <= size; i++)  
 if (m\_figure[i] != nullptr)  
 delete m\_figure[i];  
 delete[] m\_figure;  
}

**main.cpp:**

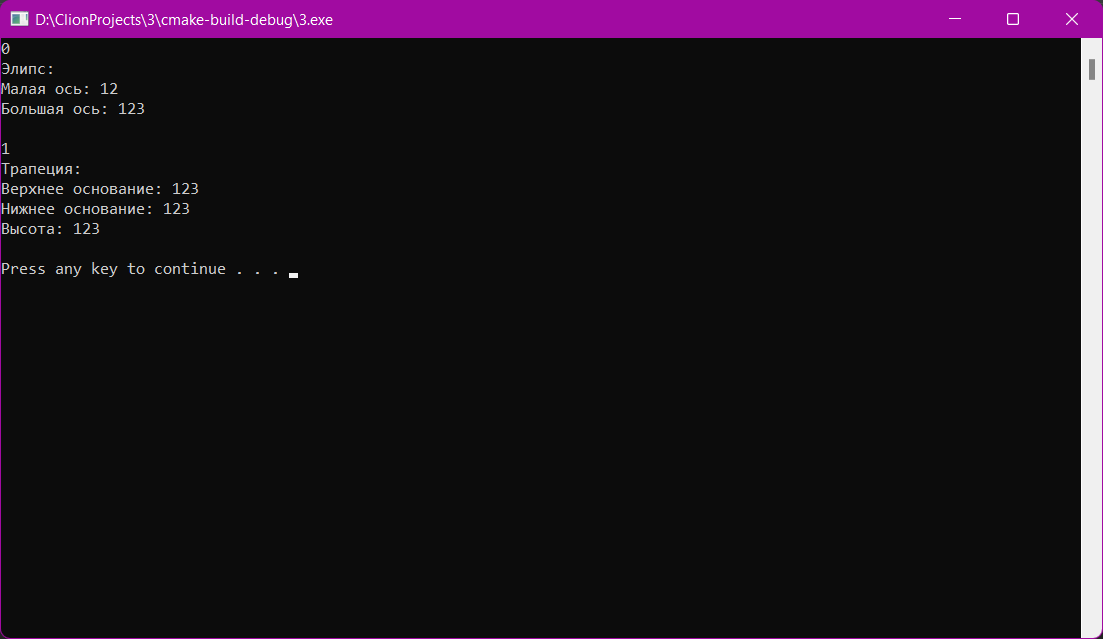
#include "LFigure.h"  
  
void text();  
int menu();  
void add\_elements(Figure \*&);  
void comparison();  
  
  
int main() {  
 SetConsoleCP(CP\_UTF8);  
 SetConsoleOutputCP(CP\_UTF8);  
 text();  
 cout << "Введите размер массива: ";  
 int size, c;  
 cin >> size;  
 system("cls");  
 LFigure mas(size);  
 Figure \*f;  
 while ((c = menu())) {  
 try {  
 switch (c) {  
 case 1:  
 add\_elements(f);  
 mas.add(f);  
 break;  
 case 2:  
 comparison();  
 break;  
 case 3:  
 mas.truncation();  
 break;  
 case 4:  
 mas.val();  
 break;  
 case 5:  
 mas.sum();  
 break;  
 case 6:  
 add\_elements(f);  
 mas.ins(f);  
 break;  
 case 7:  
 mas.del\_middle();  
 break;  
 case 8:  
 mas.show();  
 break;  
 }  
 system("pause");  
 system("cls");  
 }  
 catch (int) {  
 cout << "\nЭлемента под таким индексом нет" << endl;  
 system("pause");  
 system("cls");  
 }  
 catch (LFigure) {  
 cout << "\nВыход за пределы массива!" << endl;  
 system("pause");  
 system("cls");  
 }  
 catch (double) {  
 cout << "\nСлишком большое число!" << endl;  
 system("pause");  
 system("cls");  
 }  
 }  
 return 0;  
}  
  
void text() {  
 cout << "Написать программу, в которой описана иерархия классов: геометрические финуры" << endl  
 << "(элипс, квадрат, трапеция). Описать класс для хранения колекции финур (массива " << endl  
 << "указателей на базовый класс), в котором перегрузить операцию [], а также реализовать" << endl  
 << "функции полсчёта общей площади и периметра. Для базового класса и его потомков" << endl  
 << "перегрущить операции < == >, < != >, < = >. Продемострировать работу операторов." << endl;  
 system("pause");  
 system("cls");  
}  
  
int menu() {  
 int c;  
 cout << "1. Добавить объект класса" << endl  
 << "2. Сравнить два элемента" << endl  
 << "3. Усечение массива" << endl  
 << "4. Вывести площадь или периметр элемента" << endl  
 << "5. Вывести сумарную площадь или периметр" << endl  
 << "6. Вставка элемента" << endl  
 << "7. Удаление среднего элемента" << endl  
 << "8. Вывести массив" << endl  
 << "0. Выход из программы" << endl;  
 cout << "Выберите команду: ";  
 cin >> c;  
 system("cls");  
 return c;  
}  
  
void add\_elements(Figure \*&f) {  
 cout << "1. Элипс" << endl  
 << "2. Квадрат" << endl  
 << "3. Трапеция" << endl  
 << "Выберите класс: ";  
 int c;  
 cin >> c;  
 switch (c) {  
 case 1:  
 f = new Ellipses();  
 break;  
 case 2:  
 f = new Foursquare();  
 break;  
 case 3:  
 f = new Trapezoid();  
 break;  
 }  
}  
  
void comparison() {  
 cout << "Выберите класс: " << endl  
 << "1. Элипс" << endl  
 << "2. Квадрат" << endl  
 << "3. Трапеция" << endl  
 << "Выберите команду: ";  
 int c;  
 cin >> c;  
 switch (c) {  
 case 1: {  
 cout << "\nПервый элемент: ";  
 Ellipses o1;  
 cout << "\nВторой элемент: ";  
 Ellipses o2;  
 if (o1 == o2)  
 cout << "Элипсы равны" << endl;  
 if (o1 != o2)  
 cout << "Элипсы не равны" << endl;  
 break;  
 }  
 case 2: {  
 cout << "\nПервый элемент: ";  
 Foursquare o1;  
 cout << "\nВторой элемент: ";  
 Foursquare o2;  
 if (o1 == o2)  
 cout << "Квадраты равны" << endl;  
 if (o1 != o2)  
 cout << "Квадраты не равны" << endl;  
 break;  
 }  
 case 3: {  
 cout << "\nПервый элемент: ";  
 Trapezoid o1;  
 cout << "\nВторой элемент: ";  
 Trapezoid o2;  
  
 if (o1 == o2)  
 cout << "Трапеции равны" << endl;  
 if (o1 != o2)  
 cout << "Трапеции не равны" << endl;  
 break;  
 }  
 }  
}

**Результат программы:**

**Стартовое меню:**



**Вывод списка:**



**Суммарная площадь:**

