

ГУАП

КАФЕДРА № 14

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

ассистент

должность, уч. степень, звание

подпись, дата

Н.Ю. Чумакова

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

ВИРТУАЛЬНЫЕ ФУНКЦИИ

по курсу: ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

1042

подпись, дата

Д.А. Васейко

инициалы, фамилия

Санкт-Петербург 2022

1. Постановка задачи

Вариант 3: Создать абстрактный класс «Кривые» для вычисления координаты y для некоторой x . Создать производные классы «Прямая», «Эллипс», «Гипербола» со своими функциями вычисления y в зависимости от входного параметра x .

2. Формализация задачи

Данная программа разбита на восемь файлов: файл родительского абстрактного класса (Part.h), два файла дочернего класса «Прямая» (Straight.h и Straight.cpp), два файла дочернего класса «Эллипс» (Ellips.h и Ellips.cpp), два файла дочернего класса «Гипербола» (Hyperbola.h и Hyperbola.cpp), и один главный (main.cpp) – это файлы с объявлением, определением методов класса соответственно, а также файл с управляющей функцией.

В каждом дочернем классе есть свои приватные переменные – это координаты и коэффициенты для каждой кривой соответственно. Все они объявлены типом float. Для прямой: координаты X , Y , коэффициенты K и B : координата Y находится по уравнению « $y = kx + b$ ». Для эллипса: координаты X , Y , полуоси A и B (причем A и B должны быть больше нуля): координата Y находится по уравнению « $y = b \cdot \sqrt{(1 - x/a)(1 + 1/a)}$ ». Для гиперболы: координаты X , Y , коэффициенты K и B : координата Y находится по уравнению « $y = k/x + b$ » (причем X не должен быть равен нулю). Каждое исключение в виде недопустимых значений обрабатывается, и вместо введенного значения, переменной присваивается единица.

Пользователю доступно выбрать количество классов наследников.

В абстрактном классе Part наличествует чистая виртуальная функция float findY, которая переопределяется в каждом дочернем классе в зависимости от нужного результата, виртуальная функция print() для вывода данных на экран, а также виртуальные функции записи и считывания файла write() и read() соответственно. Также существует виртуальная функция get_id(). Все дочерние классы наследуют Part публично.

Id элемента (прямая, эллипс, гипербола) варьируется от 1 до 3 соответственно.

3. Исходный код

Файл Parr.h

```
#pragma once
#include <fstream>
class Parr {
public:
    static int count;
    Parr() {};
    ~Parr() {};
    virtual void print() = 0;
    virtual float findY() = 0;
    virtual int write(std::ofstream& outFile) = 0;
    virtual int read(std::ifstream& inFile) = 0;
    virtual int get_id() = 0;
};
```

Файл Straight.h

```
#pragma once
#include <fstream>
#include "Parr.h"
//y=kx+b
class Straight final : public Parr {
private:
    const int id = 1;
    float y = 0.0;
    float x;
    float k;
    float b;

public:
    Straight();
    Straight(float x_var, float k_var = 1.0, float b_var = 0.0);
    Straight(const Straight& copy_straight);
    ~Straight();

    static int getCount();

    float findY() override;
    void print() override;
    int write(std::ofstream& outFile);
    int read(std::ifstream& inFile);
    int get_id() override;

    float get_Y();
    float get_X();
    float get_K();
    float get_B();

    void set_Y(float Y);
    void set_X(float X);
    void set_K(float K);
    void set_B(float B);
protected:
```

```
};
```

Файл Straight.cpp

```
#include <iostream>
#include <cmath>
#include <locale>
#include "Straight.h"
#include <fstream>
```

```
using namespace std;
```

```
Straight::Straight() : x(0.0), y(0.0), k(1.0), b(0.0) { count++; }
```

```
Straight::~~Straight()
```

```
{
    x = 0;
    y = 0;
    k = 0;
    b = 0;
    count--;
}
```

```
Straight::Straight(float x_var, float k_var, float b_var)
```

```
{
    x = x_var;
    k = k_var;
    b = b_var;
    count++;
}
```

```
Straight::Straight(const Straight& copy_straight)
```

```
{
    x = copy_straight.x;
    y = copy_straight.y;
    k = copy_straight.k;
    b = copy_straight.b;
    count++;
}
```

```
float Straight::findY()
```

```
{
    y = k * x + b;
    return y;
}
```

```
float Straight::get_Y() { return y; }
```

```
float Straight::get_X() { return x; }
```

```
float Straight::get_K() { return k; }
```

```
float Straight::get_B() { return b; }
```

```
int Straight::get_id() { return id; }
```

```
void Straight::set_Y(float Y) { y = Y; }
```

```
void Straight::set_X(float X) { x = X; }
```

```
void Straight::set_B(float B) { b = B; }
```

```
void Straight::set_K(float K) { k = K; }
```

```
int Straight::write(std::ofstream& outFile)
```

```
{
    if (outFile.is_open())
    {
        outFile << this->get_id() << " " << this->get_X() << " " << this->get_K() << "
" << this->get_B() << "\n";
        return 1;
    }
}
```

```

        return 0;
    }

    int Straight::read(ifstream& inFile){
        float x_str, k_str, b_str;
        if (inFile.is_open())
        {
            inFile >> x_str >> k_str >> b_str;
            this->set_X(x_str);
            this->set_K(k_str);
            this->set_B(b_str);
            return 1;
        }
        return 0;
    }

    void Straight::print()
    {
        cout << "Координата Y прямой: " << this->findY() << ", при введенных X = " << this->get_X() << ", K = " << this->get_K() << ", B = " << this->get_B() << endl;
    }

    int Straight::getCount() {
        return count;
    }
}

```

Файл Ellips.h

```
#pragma once
```

```

#include"Parr.h"
//y=sqrt((1-(x^x)/(a^a))(b^b))
class Ellips final : public Parr {
private:
    const int id = 2;
    float x;
    float y = 0.0;
    float a;
    float b;
public:
    Ellips();
    Ellips(float x_var, float a_var = 1.0, float b_var = 1.0); //круг по умолчанию
    Ellips(const Ellips& copy_ellips);
    ~Ellips();

    float findY() override;
    void print() override;
    int write(std::ofstream& outFile);
    int read(std::ifstream& inFile);
    int get_id() override;

    float get_Y();
    float get_X();
    float get_A();
    float get_B();

    void set_Y(float Y);
    void set_X(float X);
    void set_A(float A);
    void set_B(float B);
};

```

Файл Ellips.cpp

```

#include "Ellips.h"
#include <iostream>
#include <fstream>
#include <cmath>
#include <locale>

using namespace std;

Ellips::Ellips() : x(1.0), y(0.0), a(1.0), b(1.0) { count++; }
Ellips::~Ellips() { count--; }

Ellips::Ellips(float x_var, float a_var, float b_var)
{
    a = a_var;
    b = b_var;
    x = x_var;
    count++;
}

Ellips::Ellips(const Ellips& copy_ellips)
{
    x = copy_ellips.x;
    y = copy_ellips.y;
    a = copy_ellips.a;
    b = copy_ellips.b;
    count++;
}

float Ellips::findY()
{
    float sqrt_num = (1 - x / a) * (1 + x / a);
    if (sqrt_num < 0)
        return y = -1;
    y = b * sqrt(sqrt_num);
    return y;
}

float Ellips::get_Y() { return y; }
float Ellips::get_X() { return x; }
float Ellips::get_A() { return a; }
float Ellips::get_B() { return b; }
int Ellips::get_id() { return id; }

void Ellips::set_Y(float Y) { y = Y; }
void Ellips::set_X(float X) { x = X; }
void Ellips::set_A(float A)
{
    if (A == 0)
    {
        setlocale(LC_ALL, "Rus");
        cout << "Значение A для эллипса не может быть нулевым, поэтому A будет равен
единице" << endl;
        A = 1;
    }
    a = A;
}

void Ellips::set_B(float B)
{
    if (B == 0)
    {
        setlocale(LC_ALL, "Rus");
        cout << "Значение B для эллипса не может быть нулевым, поэтому B будет равен
единице" << endl;
        B = 1;
    }
}

```

```

    }
    b = B;
}

int Ellips::write(std::ofstream& outFile)
{
    if (outFile.is_open())
    {
        outFile << this->get_id() << " " << this->get_X() << " " << this->get_A() << "
" << this->get_B() << "\n";
        return 1;
    }
    return 0;
}

int Ellips::read(std::ifstream& inFile)
{
    float x_elp, a_elp, b_elp;

    inFile >> x_elp >> a_elp >> b_elp;

    if (inFile.is_open())
    {
        if (a_elp == 0)
        {
            setlocale(LC_ALL, "Rus");
            cout << "Значение A для эллипса не может быть нулевым, поэтому A будет
равен единице" << endl;
            a_elp = 1;
        }
        if (b_elp == 0)
        {
            setlocale(LC_ALL, "Rus");
            cout << "Значение B для эллипса не может быть нулевым, поэтому B будет
равен единице" << endl;
            b_elp = 1;
        }

        this->set_X(x_elp);
        this->set_A(a_elp);
        this->set_B(b_elp);
        return 1;
    }
    return 0;
}

void Ellips::print()
{
    cout << "Координата Y эллипса: " << this->findY() << ", при введенных X = " << this-
>get_X() << ", A = " << this->get_A()<< ", B = " << this->get_B()<< endl;
}

```

Файл Hyperbola.h

```
#pragma once
```

```
#include "Parr.h"
```

```

class Hyperbola final : public Parr {
private:
    const int id = 3;
    float x;
    float y = 0.0;
}

```

```

        float k;
        float b;
public:
    Hyperbola();
    Hyperbola(float x_var, float k_var = 1.0, float b_var = 0);
    Hyperbola(const Hyperbola& copy_hyper);
    ~Hyperbola();

    float findY() override;
    void print() override;
    int write(std::ofstream& outFile);
    int read(std::ifstream& inFile);
    int get_id() override;

    float get_Y();
    float get_X();
    float get_K();
    float get_B();

    void set_Y(float Y);
    void set_X(float X);
    void set_K(float K);
    void set_B(float B);
};

```

Файл Hyperbola.cpp

```

#include "Hyperbola.h"
#include <iostream>
#include <cmath>
#include <locale>

using namespace std;

Hyperbola::Hyperbola() : x(1.0), y(0.0), k(1.0), b(0.0) { count++; }
Hyperbola::~Hyperbola() { count--; }

Hyperbola::Hyperbola(float x_var, float k_var, float b_var)
{
    x = x_var;
    k = k_var;
    b = b_var;
    count++;
}

Hyperbola::Hyperbola(const Hyperbola& copy_hyper)
{
    x = copy_hyper.x;
    y = copy_hyper.y;
    k = copy_hyper.k;
    b = copy_hyper.b;
    count++;
}

float Hyperbola::findY()
{
    y = k / x + b;
    return y;
}

float Hyperbola::get_Y() { return y; }
float Hyperbola::get_X() { return x; }

```



```

float Hyperbola::get_K() { return k; }
float Hyperbola::get_B() { return b; }
int Hyperbola::get_id() { return id; }

void Hyperbola::set_Y(float Y) { y = Y; }
void Hyperbola::set_X(float X)
{
    if (X == 0)
    {
        setlocale(LC_ALL, "Rus");
        cout << "Значение X для гиперболы не может быть нулевым, поэтому X будет равен
единице" << endl;
        X = 1;
    }
    x = X;
}
void Hyperbola::set_B(float B) { b = B; }
void Hyperbola::set_K(float K) { k = K; }

int Hyperbola::write(std::ofstream& outFile)
{
    if (outFile.is_open())
    {
        outFile << this->get_id() << " " << this->get_X() << " " << this->get_K() << "
" << this->get_B() << "\n";
        return 1;
    }
    return 0;
}

int Hyperbola::read(ifstream& inFile) {
    float x_hyp, k_hyp, b_hyp;
    if (inFile.is_open())
    {
        inFile >> x_hyp >> k_hyp >> b_hyp;
        if (x_hyp == 0)
        {
            setlocale(LC_ALL, "Rus");
            cout << "Значение X для гиперболы не может быть нулевым, поэтому X
будет равен единице" << endl;
            x_hyp = 1;
        }

        this->set_X(x_hyp);
        this->set_K(k_hyp);
        this->set_B(b_hyp);
        return 1;
    }
    return 0;
}

void Hyperbola::print()
{
    cout << "Координата Y гиперболы: " << this->findY() << ", при введенных X = " <<
this->get_X() << ", K = " << this->get_K() << ", B = " << this->get_B() << endl;
}

```

Файл main.cpp

```

#include <iostream>
#include <cmath>
#include <locale>
#include <string>
#include "Parr.h"

```

```

#include "Straight.h"
#include "Ellips.h"
#include "Hyperbola.h"
#include <fstream>
using namespace std;

/*Вариант 3 - Создать абстрактный класс "Кривые" для вычисления координаты у для некоторой
х.
Создать производные классы "Прямая", "Эллипс", "Гипербола" со своими функциями вычисления у
в зависимости от входного параметра х*/

void readFile(Parr**& arr, int& N);
void AddNew_ffile(Parr**& arr, int& N, char id = 0);
void AddNew(Parr**& arr, int& N, char id);
void printArr(Parr**& arr, int &N);
void writeFile(Parr**& arr, int& N);
int Straight::count = 0;

int main()
{
    setlocale(LC_ALL, "Rus");

    int c, exit_equ = 0;
    int N = 0;
    Parr** parr = new Parr * [N];

    cout << "Вас приветствует программа 3-ей лабораторной по ТП!" << endl;
    cout << "Хотите ли вы ввести данные из файла?" << endl;
    cout << "1 - Да" << endl;
    cout << "Any key - Нет" << endl;
    cout << "--> ";
    cin >> c;
    system("cls");
    if (c == 1)
    {
        readFile(parr, N);
        if (N == 0)
        {
            cout << "Файл пуст" << endl;
        }
    }
    printArr(parr, N);
    while (exit_equ != 1)
    {
        cout << endl;
        cout << "Что вы хотите добавить" << endl;
        cout << "1 - Прямую" << endl;
        cout << "2 - Эллипс" << endl;
        cout << "3 - Гиперболу" << endl;
        cout << "0 - Выход" << endl;
        cout << "--> ";
        cin >> c;
        switch (c)
        {
            case 1:
                AddNew(parr, N, 1);
                break;
            case 2:
                AddNew(parr, N, 2);
                break;
            case 3:

```

```

        AddNew(parr, N, 3);
        break;
    case 0:
        exit_equ = 1;
        break;
    }
    printArr(parr, N);
}
cout << endl;
cout << "Хотите ли вы внести данные в файл?" << endl;
cout << "1 - Да" << endl;
cout << "Any key - Нет" << endl;
cout << "--> ";
cin >> c;
if (c == 1)
{
    writeFile(parr, N);
}
getchar();
return 0;
}

void AddNew_ffile(Parr**& arr, int& N, char id)
{
    Parr** newarr = new Parr * [N+1];
    for (int i = 0; i < N; i++)
        newarr[i] = arr[i];
    switch (id)
    {
    case 1:
        newarr[N] = new Straight(N, N, N );
        break;
    case 2:
        newarr[N] = new Ellips(N, N, N);
        break;
    case 3:
        newarr[N] = new Hyperbola(N, N, N);
        break;
    }

    N++;

    delete [] arr;
    arr = newarr;
}

void readFile(Parr**& arr, int& N)
{
    int n, id;
    ifstream inFile;
    string stra =
"C:\\Flesh\\SUAI_works\\T0\\ThirdLab\\ThirdLab\\ThirdLab\\straight.txt";
    inFile.open(stra, ios_base::in);
    if (!inFile.is_open())
    {
        cout << "Error" << endl;
        return;
    }
    inFile >> n;
    for (int i=0; i < n; i++) {
        inFile >> id;
        AddNew_ffile(arr, N, id);
        arr[N-1]->read(inFile);
    }
}

```

```

    }
    inFile.close();
}

void writeFile(Parr**& arr, int& N) {
    ofstream outFile;
    string str =
"C:\\Flesh\\SUAI_works\\T0\\ThirdLab\\ThirdLab\\ThirdLab\\straight.txt";
    outFile.open(str, ios_base::out);
    if (!outFile.is_open())
    {
        cout << "Error" << endl;
        return;
    }
    outFile << Straight::getCount() << "\n";
    for (int i = 0; i < N; i++){
        arr[i]->write(outFile);
    }
    outFile.close();
}

void AddNew(Parr**& arr, int& N, char id)
{
    Parr** newarr = new Parr * [N + 1];
    for (int i = 0; i < N; i++){
        newarr[i] = arr[i];
    }
    switch (id)
    {
    case 1:
        float x_str, k_str, b_str; //straight
        system("cls");
        cout << "Введите коэффициенты для прямой через пробел (x, k, b): ";
        cin >> x_str >> k_str >> b_str;
        newarr[N] = new Straight(x_str, k_str, b_str);
        break;
    case 2:
        float x_ell, a_ell, b_ell; //ellips
        system("cls");
        cout << "Введите коэффициенты для эллипса через пробел (x, a, b): ";
        cin >> x_ell >> a_ell >> b_ell;
        if (a_ell == 0)
        {
            cout << "Значение А для эллипса не может быть нулевым, поэтому А будет
равен единице" << endl;
            a_ell = 1;
        }
        if (b_ell == 0)
        {
            cout << "Значение В для эллипса не может быть нулевым, поэтому В будет
равен единице" << endl;
            b_ell = 1;
        }
        newarr[N] = new Ellips(x_ell, a_ell, b_ell);
        break;
    case 3:
        float x_hyp, k_hyp, b_hyp; //hyperbola
        system("cls");
        cout << "Введите коэффициенты для гиперболы через пробел (x, k, b): ";
        cin >> x_hyp >> k_hyp >> b_hyp;
        if (x_hyp == 0)
        {

```

```

        cout << "Значение X для гиперболы не может быть нулевым, поэтому X будет равен
единице" << endl;
        x_hyp = 1;
    }
    newarr[N] = new Hyperbola(x_hyp, k_hyp, b_hyp);
    break;
}
newarr[N]->print();
N++;

delete[] arr;
arr = newarr;
}

void printArr(Parr**& arr, int& N)
{
    for (int i = 0; i < N; i++)
    {
        cout << "Array el >> " << i + 1 << " << ";
        arr[i]->print();
    }
}

```

4. Результаты работы программы

Для демонстрации работы введем три примера: с отрицательными значениями, с нулем у эллипса, с нулем у гиперболы. Отрицательные числа введем в файл, а нули будем вводить вручную. Одновременно с этим продемонстрируем возможность перезаписи файла.

Пример первый. Значения прямой (X, K, B): -10, 5.5, 8. Значения эллипса (X, A, B): -1, 4.33, 6.99. Значения гиперболы (X, K, B): 1, 2, 3.

На рисунке 1 изображен открытый текстовый файл с данными.

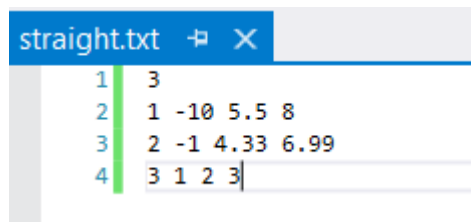


Рисунок 1 – Текстовый файл с данными

Программа встречает нас следующим экраном (рис. 2)

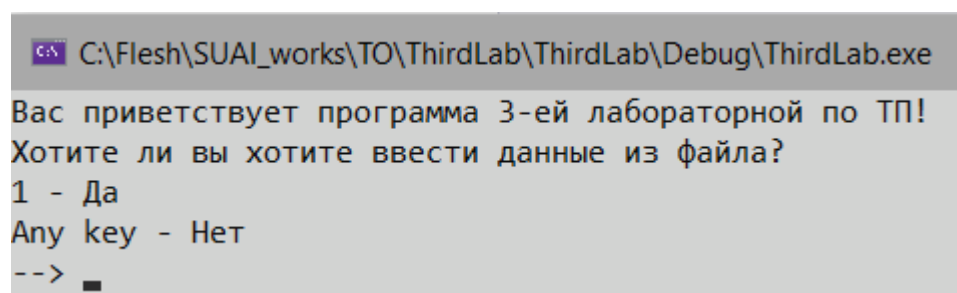


Рисунок 2 – Активация программы

Выбираем ввод из файла. Ввод завершен успешно. (рис. 3)

```
C:\Flesh\SUAI_works\TO\ThirdLab\ThirdLab\Debug\ThirdLab.exe
Array el >> 1 << Координата Y прямой: -47, при введенных X = -10, K = 5.5, B = 8
Array el >> 2 << Координата Y эллипса: 6.80103, при введенных X = -1, A = 4.33, B = 6.99
Array el >> 3 << Координата Y гиперболы: 5, при введенных X = 1, K = 2, B = 3

Что вы хотите добавить
1 - Прямую
2 - Эллипс
3 - Гиперболу
0 - Выход
-->
```

Рисунок 3 – Взятие данных из файла

Выбираем прямую. (Рис. 4)

```
C:\Flesh\SUAI_works\TO\ThirdLab\ThirdLab\Debug\ThirdLab.exe
Введите коэффициенты для прямой через пробел (x, k, b): 1 3 5.5
Координата Y прямой: 8.5, при введенных X = 1, K = 3, B = 5.5
```

Рисунок 4 -Выбираем прямую

Выбираем эллипс. (рис. 5)

```
C:\Flesh\SUAI_works\TO\ThirdLab\ThirdLab\Debug\ThirdLab.exe
Введите коэффициенты для эллипса через пробел (x, a, b): 4 0 0
Значение A для эллипса не может быть нулевым, поэтому A будет равен единице
Значение B для эллипса не может быть нулевым, поэтому B будет равен единице
Координата Y эллипса: -1, при введенных X = 4, A = 1, B = 1
```

Рисунок 5 – Выбираем эллипс

Выбираем гиперболу (рис. 6)

```
C:\Flesh\SUAI_works\TO\ThirdLab\ThirdLab\Debug\ThirdLab.exe
Введите коэффициенты для гиперболы через пробел (x, k, b): 0 3.5 1
Значение X для гиперболы не может быть нулевым, поэтому X будет равен единице
Координата Y гиперболы: 4.5, при введенных X = 1, K = 3.5, B = 1
```

Рисунок 6 – Выбираем гиперболу

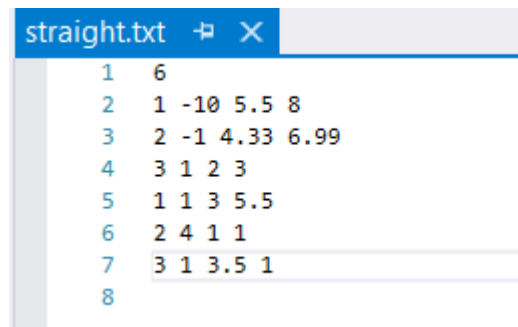
Выбираем выход из программы. (Рис. 7)

```
Array el >> 1 << Координата Y прямой: -47, при введенных X = -10, K = 5.5, B = 8
Array el >> 2 << Координата Y эллипса: 6.80103, при введенных X = -1, A = 4.33, B = 6.99
Array el >> 3 << Координата Y гиперболы: 5, при введенных X = 1, K = 2, B = 3
Array el >> 4 << Координата Y прямой: 8.5, при введенных X = 1, K = 3, B = 5.5
Array el >> 5 << Координата Y эллипса: -1, при введенных X = 4, A = 1, B = 1
Array el >> 6 << Координата Y гиперболы: 4.5, при введенных X = 1, K = 3.5, B = 1

Хотите ли вы хотите внести данные в файл?
1 - Да
Any key - Нет
--> █
```

Рисунок 7 – выход из программы.

Сохраняем. (Рис. 8)



1	6				
2	1	-10	5.5	8	
3	2	-1	4.33	6.99	
4	3	1	2	3	
5	1	1	3	5.5	
6	2	4	1	1	
7	3	1	3.5	1	
8					

Рисунок 8 – изменения в файле после сохранения

5. Выводы

В процессе выполнения лабораторной работы мы изучили основы применения виртуальных функций, вспомнили наследование классов. Изучили принципы работы виртуальных функций через таблицы виртуальных функций; продемонстрировали их применение через создание указателя родительского класса на объект дочернего класса.