|  |  |
| --- | --- |
| для прик эмбл | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждениевысшего профессионального образования"Московский технологический университет"МИРЭА | |
|  | Факультет информационных технологий (ИТ) |
|  | Кафедра практической и прикладной информатики (ППИ) |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ** | |
| **по дисциплине** | |
| «Объектно-ориентированное программирование» | |
|  | |
| Выполнил студент группы ИНБО-04-18 | Кузин М.В. |
| Принял старший преподаватель | Хлебникова В.Л. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |

Москва 2019

Оглавление

[Оглавление 2](#_Toc8582741)

[Таблица успеваемости 3](#_Toc8582742)

[Практическая работа №1 4](#_Toc8582743)

[Практическая работа №2 6](#_Toc8582744)

[Практическая работа №3 12](#_Toc8582745)

[Практическая работа №4 16](#_Toc8582746)

[Практическая работа №5 21](#_Toc8582747)

[Практическая работа №6 24](#_Toc8582748)

[Практическая работа №7 35](#_Toc8582749)

[Практическая работа №8 42](#_Toc8582750)

# Таблица успеваемости

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/р. | страница в отчете. | Студент выполнил:  (подпись) | Преподаватель принял:  (подпись) | Балл  (максимально возможный) | Балл (Фактический) |
| 1. |  |  |  |  |  |
| 2. |  |  |  |  |  |
| 3. |  |  |  |  |  |
| 4. |  |  |  |  |  |
| 5. |  |  |  |  |  |
| 6. |  |  |  |  |  |
| 7. |  |  |  |  |  |
| 8. |  |  |  |  |  |
|  |  |  | Итоговый балл: |  |  |

# Практическая работа №1

**Задание:**

1.Реализовать программу, которая считывает Ваше имя и

здоровается с Вами.

2.Реализовать программу, производящую различные операции над числами. (сложение\вычитание, целые числа\числа с плавающей запятой)

**Ход выполнения:**

|  |
| --- |
| #include <iostream> |
|  | #include "string" |
|  | using namespace std; |
|  | int main() |
|  | { |
|  | setlocale(LC\_ALL, "Rus"); |
|  | string name; |
|  | cout << "Введите ваше имя : "; |
|  | cin >> name; |
|  | cout << "Добрый день, " << name << "!!!" << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |
| #include <iostream> |
|  | using namespace std; |
|  | int main() |
|  | { |
|  | setlocale(LC\_ALL, "Rus"); |
|  | double a; |
|  | double b; |
|  | cout << "Введите число a : "; |
|  | cin >> a; |
|  | cout << "Введите число b : "; |
|  | cin >> b; |
|  | cout << "сумма а и b = " << a + b << endl; |
|  | cout << "разность а и b = " << a - b << endl; |
|  | cout << "произведение а и b = " << a \* b << endl; |
|  | cout << "частное а и b = " << a / b << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Практической работы №1 были использованы библиотеки iostream, string, windows.h. Были достигнуты результаты:

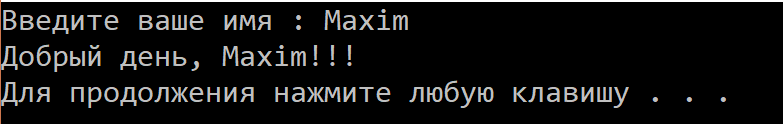


Рис. 1

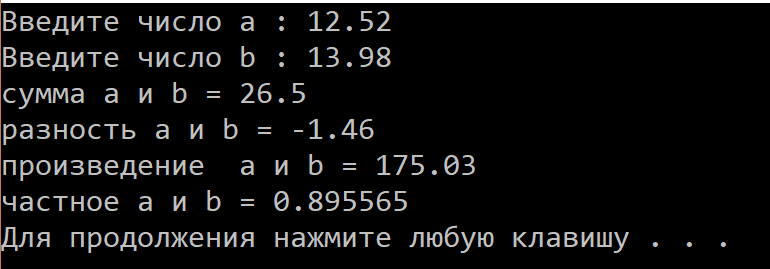


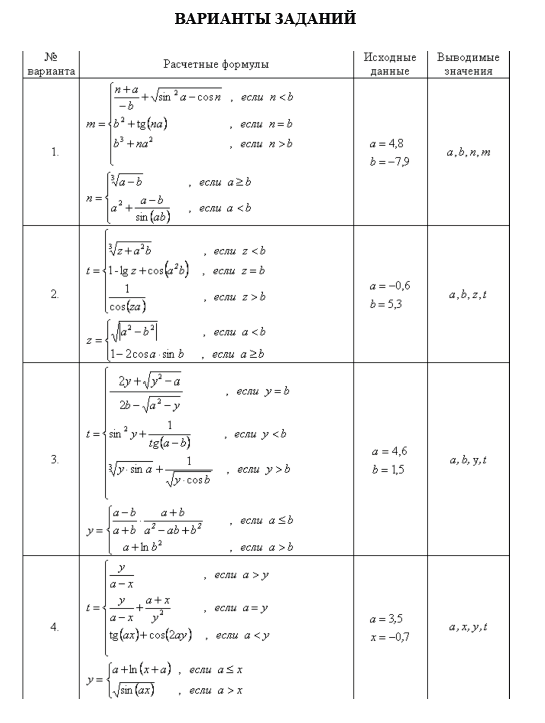
Рис. 2

<https://github.com/FunFuhrer/OOP/blob/master/practika11.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika1.2.cpp>

# Практическая работа №2

**Задание:**



**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include <iostream> |
|  | using namespace std; |
|  | int main() |
|  | { |
|  | double a; |
|  | double b; |
|  | double n; |
|  | double m; |
|  | cout << "a="; |
|  | cin >> a; |
|  | cout << "b="; |
|  | cin >> b; |
|  |  |
|  | if (a >= b) |
|  | n=pow(a-b, 1. / 3.); |
|  | else |
|  | n = pow(a, 2) + (a - b) / sin(a\*b); |
|  |  |
|  | if (n < b) |
|  | m = (n + a) / (-b) + pow(pow(sin(a), 2) - cos(n), 1. / 2.); |
|  | else if (n == b) |
|  | m = pow(b, 2) + tan(n\*a); |
|  | else |
|  | m = pow(b, 3) + n \* pow(a, 2); |
|  | cout << "a=" << a << endl; |
|  | cout << "b=" << b << endl; |
|  | cout << "n=" << n << endl; |
|  | cout << "m=" << m << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 2

|  |
| --- |
| #include <iostream> |
|  | using namespace std; |
|  | int main() |
|  | { |
|  | double a; |
|  | double b; |
|  | double z; |
|  | double t; |
|  | cout << "a="; |
|  | cin >> a; |
|  | cout << "b="; |
|  | cin >> b; |
|  |  |
|  | if (a >= b) |
|  | z = 1 - 2 \* cos(a)\*sin(b); |
|  | else |
|  | z = pow(abs(pow(a,2)-pow(b,2)), 1. / 2.); |
|  |  |
|  | if (z < b) |
|  | t = pow((z + pow(a, 2)\*b), 1. / 3.); |
|  | else if (z == b) |
|  | t = 1 - log10(z) + cos(pow(a, 2)\*b); |
|  | else |
|  | t = 1 / (cos(z\*a)); |
|  | cout << "a=" << a << endl; |
|  | cout << "b=" << b << endl; |
|  | cout << "z=" << z << endl; |
|  | cout << "t=" << t << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 3

|  |
| --- |
| #include <iostream> |
|  | using namespace std; |
|  | int main() |
|  | { |
|  | double a; |
|  | double b; |
|  | double y; |
|  | double t; |
|  | cout << "a="; |
|  | cin >> a; |
|  | cout << "b="; |
|  | cin >> b; |
|  |  |
|  | if (a > b) |
|  | y = a+2\*log(b); |
|  | else |
|  | y = ((a-b)/(a+b))\*((a+b)/(a\*a-a\*b+b\*b)); |
|  |  |
|  | if (y < b) |
|  | t = sin(y)\*sin(y) + 1 / (tan(a - b)); |
|  | else if (y == b) |
|  | t = (2 \* y + pow(y\*y - a, 1. / 2.)) / (2 \* b - pow(a\*a - y, 1. / 2.)); |
|  | else |
|  | t = pow(y\*sin(a), 1. / 3.) + 1 / (pow(y\*cos(b), 1. / 2.)); |
|  | cout << "a=" << a << endl; |
|  | cout << "b=" << b << endl; |
|  | cout << "y=" << y << endl; |
|  | cout << "t=" << t << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 4

|  |
| --- |
| #include <iostream> |
|  | using namespace std; |
|  | int main() |
|  | { |
|  | double a; |
|  | double x; |
|  | double y; |
|  | double t; |
|  | cout << "a="; |
|  | cin >> a; |
|  | cout << "x="; |
|  | cin >> x; |
|  |  |
|  | if (a > x) |
|  | y = pow(sin(a\*x), 1. / 2.); |
|  | else |
|  | y = a + log(x + a); |
|  | if (y > a) |
|  | t = tan(a\*x) + cos(2 \* a\*y); |
|  | else if (y == a) |
|  | t = y / (a - x) + (a + x) / (y\*y); |
|  | else |
|  | t = y / (a - x); |
|  | cout << "a=" << a << endl; |
|  | cout << "x=" << x << endl; |
|  | cout << "y=" << y << endl; |
|  | cout << "t=" << t << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Практической работы №2 были использованы библиотеки iostream, cmath. Были достигнуты результаты:

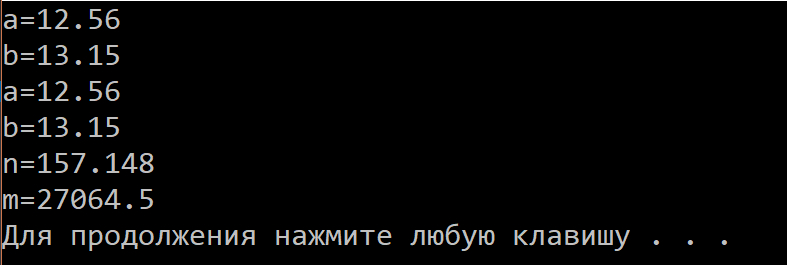


Рис. 3

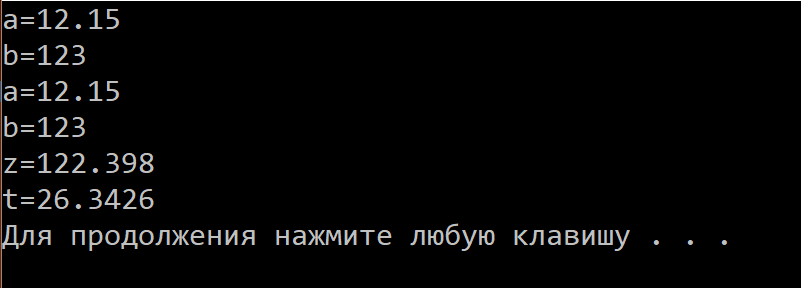


Рис. 4

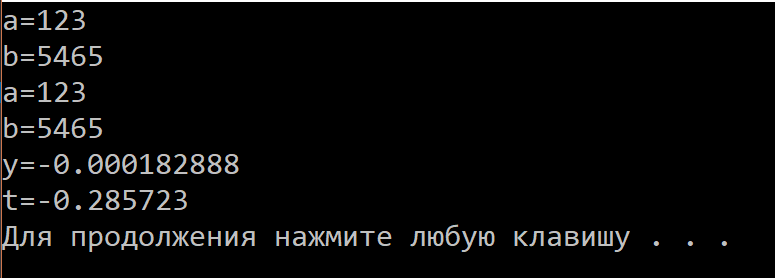


Рис. 5

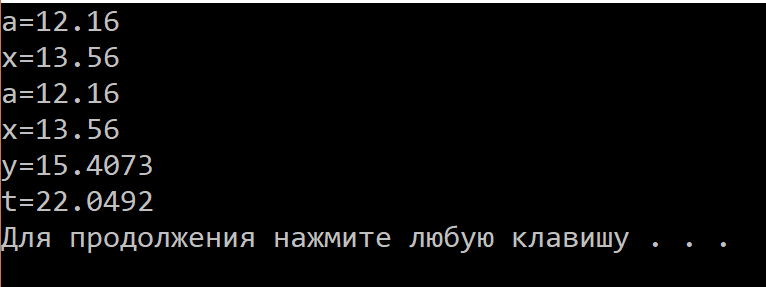


Рис. 6

<https://github.com/FunFuhrer/OOP/blob/master/practika2.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika2.2.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika2.3.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika2.4.cpp>

# Практическая работа №3

**Задание:**

1. Реализовать класс «Книга»
2. Реализовать класс «Группа студентов»
3. Реализовать класс «Автомобиль»
4. Реализовать класс «Вектор»
5. Реализовать класс «Библиотека» (повышенная сложность задания)

**Ход выполнения:**

|  |
| --- |
| #include<iostream> |
|  | #include<string> |
|  | #include <vector> |
|  | using namespace std; |
|  |  |
|  | class book |
|  | { |
|  | private: |
|  | string name, author; |
|  | int year; |
|  | public: |
|  | book(string name, string author, int year) |
|  | { |
|  | this->name = name; |
|  | this->author = author; |
|  | this->year = year; |
|  | } |
|  | book(string name, string author) |
|  | { |
|  | this->name = name; |
|  | this->author = author; |
|  | } |
|  | string getName() { |
|  | return name; |
|  | } |
|  | string getAuthor() { |
|  | return author; |
|  | } |
|  | int getYear() { |
|  | return year; |
|  | } |
|  | void setName(string name) { |
|  | this->name = name; |
|  | } |
|  | void setAuthor(string author) { |
|  | this->author = author; |
|  | } |
|  | void setYear(int year) { |
|  | this->year = year; |
|  | } |
|  | }; |
|  | class Group |
|  | { |
|  | string name; |
|  | int countOfMembers, countExams; |
|  | public: |
|  | void setName(string name) { |
|  | this->name = name; |
|  | } |
|  | void setMembers(int countOfMembers) { |
|  | this->countOfMembers = countOfMembers; |
|  | } |
|  | void setExams(int countExams) { |
|  | this->countExams = countExams; |
|  | } |
|  | string getName() { |
|  |  |
|  | return name; |
|  | } |
|  | int getMembers() { |
|  | return countOfMembers; |
|  | } |
|  | int getExams() { |
|  | return countExams; |
|  | } |
|  | }; |
|  | class Auto { |
|  | string brand, model; |
|  | int year; |
|  | public: |
|  | void setBrand(string brand) { |
|  | this->brand = brand; |
|  | } |
|  | void setModel(int model) { |
|  | this->model = model; |
|  | } |
|  | void setYear(int year) { |
|  | this->year = year; |
|  | } |
|  | string getBrand() { |
|  | return brand; |
|  |  |
|  | } |
|  | string getModel() { |
|  | return model; |
|  | } |
|  | int getYear() { |
|  | return year; |
|  | } |
|  | }; |
|  | class Vector { |
|  | double module; |
|  | double x1, x2, y1, y2; |
|  | public: |
|  | Vector(double x2, double y2) |
|  | { |
|  | this->x1 = 0; |
|  | this->y1 = 0; |
|  | this->x2 = x2; |
|  | this->y2 = y2; |
|  | } |
|  | void setStartPoint(double x1, double y1) { |
|  | this->x1 = x1; |
|  | this->y1 = y1; |
|  | } |
|  | void setEndPoint(double x2, double y2) { |
|  | this->x2 = x2; |
|  | this->y2 = y2; |
|  | } |
|  |  |
|  | double getModule() { |
|  | return sqrt((x1 - x2)\*(x1 - x2) + (y1 - y2)\*(y1 - y2)); |
|  | } |
|  | }; |
|  | class Librar { |
|  | int countBooks; |
|  | vector<book> books; |
|  | public: |
|  | void setCountBooks(int countBooks) { |
|  | this->countBooks = countBooks; |
|  | } |
|  | void addBook(book a) { |
|  | books.push\_back(a); |
|  | } |
|  | int getCountBooks() { |
|  | return countBooks; |
|  | } |
|  | }; |
|  | int main() |
|  | { |
|  | setlocale(0, "rus"); |
|  | book firstBook("Мастер и Маргарита", "Булгаков"); |
|  | firstBook.setYear(1967); |
|  | cout << "Автор: " << firstBook.getAuthor() << "\nНазвание: " << firstBook.getName() << "\nГод: " << firstBook.getYear() << endl; |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Практической работы №3 были использованы библиотеки iostream, string. В классах реализованы методы для присвоения значений полям и вывода этих значений. Были достигнуты результаты:

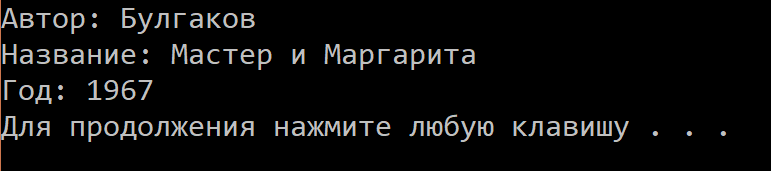


Рис. 7

<https://github.com/FunFuhrer/OOP/blob/master/practika3.cpp>

# Практическая работа №4

**Задание:**

1. Объявите указатель на массив типа double и предложите

пользователю выбрать его размер. Далее напишите четыре

функции: первая должна выделить память для массива, вторая —

заполнить ячейки данными, третья — показать данные на экран,

четвертая — освободить занимаемую память. Программа должна

предлагать пользователю продолжать работу (создавать новые

динамические массивы) или выйти из программы.

1. Объявите указатель на массив типа int и выделите память память

для 12-ти элементов. Необходимо написать функцию, которая

поменяет значения четных и нечетных ячеек массива.

1. Объявить и заполнить двумерный динамический массив

случайными числами от 10 до 50. Показать его на экран. Для

заполнения и показа на экран написать отдельные функции.

(подсказка: функции должны принимать три параметра — указатель на динамический массив, количество строк, количество столбцов). Количество строк и столбцов выбирает пользователь.

**Ход выполнения:**

Вариант 1

#include <iostream>

using namespace std;

double\* first(int n) {

double\* a = new double[n];

return a;

}

void second(double\* m, int n) {

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

{

m[i] = i;

}

}

void third(double\* m, int n) {

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

{

cout << m[i] << "\t";

}

cout << endl;

}

void fourth(double\*m) {

delete[] m;

}

int main()

{

setlocale(0, "rus");

int size = -1;

double \*arr;

while (size != 0)

{

cout << "Чтобы выйти введите 0" << endl;

cout << "Введите размер массива: ";

cin >> size;

cout << endl;

arr = first(size);

second(arr, size);

third(arr, size);

fourth(arr);

}

system("pause");

return 0;

}Вариант 2

|  |
| --- |
| #include <iostream> |
| #include <ctime> |  |
| using namespace std; |  |
| void Swap(int\* m, int n) { |  |
|  | int k; |
|  | for (int i = 0; i < n; i++) |
|  | { |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | } |
|  | cout << endl; |
| } |  |
|  |  |
| int main() |  |
| { |  |
|  | setlocale(0, "rus"); |
|  | srand(time(0)); |
|  | int size = 12; |
|  | int \*arr; |
|  | cout << endl; |
|  | arr = new int[size]; |
|  | for (int i = 0; i < size; i++) |
|  | { |
|  |  |
|  | } |
|  | cout << "Изначальный массив:\t\t"; |
|  | for (int i = 0; i < size; i++) |
|  | { |
|  |  |
|  | } |
|  | Swap(arr, size); |
|  | cout << "Массив после функции:\t\t"; |
|  | for (int i = 0; i < size; i++) |
|  | { |
|  |  |
|  | } |
|  | delete[] arr; |
|  | system("pause"); |
|  | return 0; |
| } |  |

Вариант 3

|  |
| --- |
| #include <iostream> |
|  | #include <ctime> |
|  | using namespace std; |
|  | void init(int\*\*a,int rows,int cols) { |
|  | for (int i = 0; i < rows; i++) |
|  | { |
|  | for (int j = 0; j < cols; j++) |
|  | { |
|  | a[i][j] = rand() % 41 + 10; |
|  | } |
|  | } |
|  | } |
|  | void show(int\*\*a, int rows, int cols) { |
|  | for (int i = 0; i < rows; i++) |
|  | { |
|  | for (int j = 0; j < cols; j++) |
|  | { |
|  | cout << a[i][j] << "\t"; |
|  | } |
|  | cout << endl; |
|  | } |
|  | } |
|  |  |
|  |  |
|  | int main() |
|  | { |
|  | setlocale(0, "rus"); |
|  | srand(time(0)); |
|  | int cols, rows; |
|  | cout << "Строки: \t"; |
|  | cin >> rows; |
|  | cout << endl; |
|  | cout << "Столбцы:\t"; |
|  | cin >> cols; |
|  | cout << endl; |
|  | int \*\*arr = new int\*[rows]; |
|  | for (int i = 0; i < rows; i++) |
|  | arr[i] = new int[cols]; |
|  | init(arr, rows, cols); |
|  | cout << "Массив :" << endl; |
|  | show(arr, rows, cols); |
|  |  |
|  |  |
|  | for (int i = 0; i < rows; i++) |
|  | delete[] arr[i]; |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Практической работы №4 были использованы библиотеки iostream, cmath, ctime. Были реализованы функции перестановки, заполнения, вывода массива. Были достигнуты результаты:

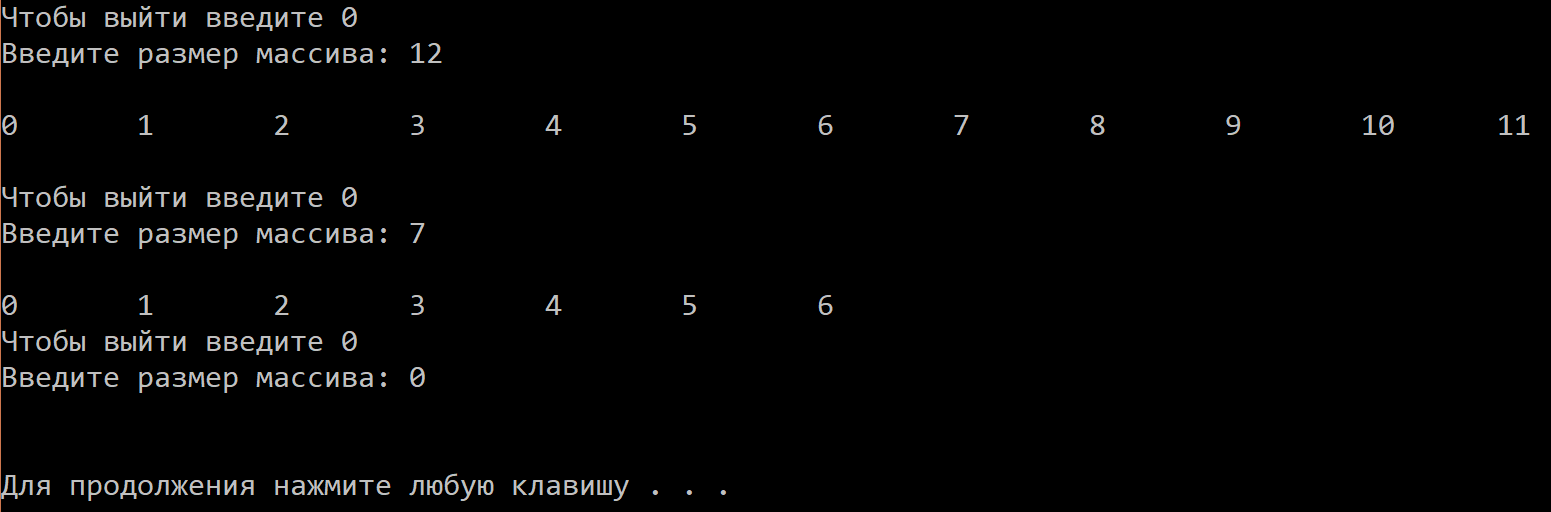


Рис. 8

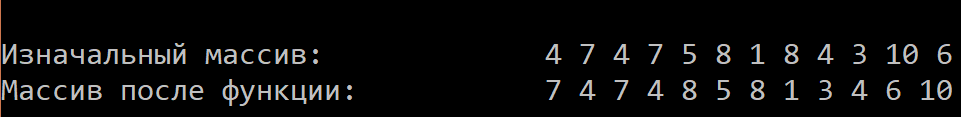


Рис. 9

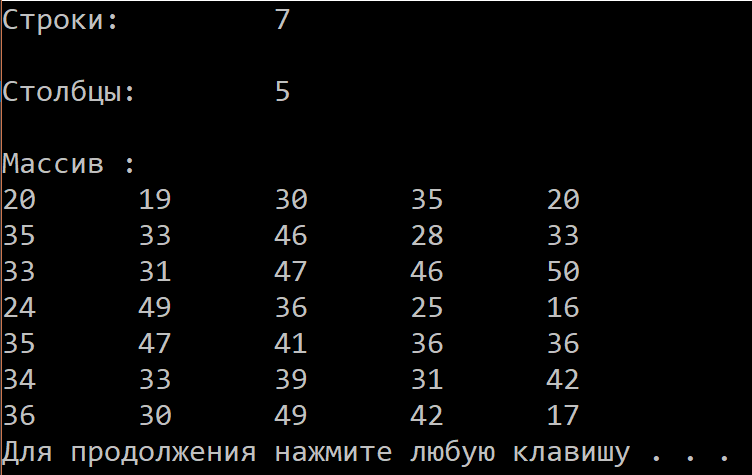


Рис. 10

<https://github.com/FunFuhrer/OOP/blob/master/practika4.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika4.2.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika4.3.cpp>

# Практическая работа №5

**Задание:**

1. Реализовать сортировку пузырьком для целых чисел, а затем

перегрузить её для дробных.

1. Реализовать сортировку выбором для целых чисел, а затем

перегрузить её для дробных.

1. Реализовать сортировку вставками для целых чисел, а затем

перегрузить её для дробных.

1. Реализовать программу-калькулятор, работающий с различными

типами данных

**Ход выполнения:**

#include <iostream>

#include <ctime>

using namespace std;

template <typename T>

void bublesort(T\* arr, int size) {

T temp;

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

cout << "Произошла сортировка пузырьком\n";

}

template <typename T>

void selectSort(T\* a, int size) {

int i, j, k;

T x;

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

k = i; x = a[i];

for (j = i + 1; j < size; j++)

if (a[j] < x) {

k = j; x = a[j];

}

a[k] = a[i]; a[i] = x;

}

cout << "Произошла сортировка выбором\n";

}

template <typename T>

void insertSort(T\* a, long size) {

T x;

long i, j;

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

x = a[i];

for (j = i - 1; j >= 0 && a[j] > x; j--)

a[j + 1] = a[j];

a[j + 1] = x;

}

cout << "Произошла сортировка вставками\n";

}

template <typename T1, typename T2>

T2 sum(T1 a, T2 b) {

return a + b;

}

template <typename T1, typename T2>

T2 diff(T1 a, T2 b) {

return a - b;

}

template <typename T1, typename T2>

T2 mul(T1 a, T2 b) {

return a \* b;

}

template <typename T1, typename T2>

T2 divide(T1 a, T2 b) {

return a / b;

}

int main()

{

srand(time(NULL));

setlocale(0, "rus");

int size;

cout << "n = ";

cin >> size;

int \* arr = new int[size];

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

arr[i] = rand() % 100;

cout << "arr[" << i << "] = " << arr[i] << "\t";

}

cout << endl << "Выбирите сортировку(1-пузырьком, 2-выбором, 3-вставками: ";

int k;

cin >> k;

switch (k)

{

case 1: {bublesort(arr, size); break; }

case 2: {selectSort(arr, size); break; }

case 3: {insertSort(arr, size); break; }

default:

break;

}

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

cout << "arr[" << i << "] = " << arr[i] << "\t";

}

delete[] arr;

cout << "Вариант 4" << endl;

int a;

double b;

cout << "Введите а и b (a-int, b-double)\n";

cin >> a >> b;

cout << sum(a, b) << ' ' << diff(a, b) << ' ' << mul(a, b) << ' ' << divide(a, b) << endl;

system("pause");

return 0;

}

**Выводы:**

В ходе выполнения Практической работы №5 были использованы библиотеки iostream, cmath, ctime. Были реализованы функции сортировок массива и функции работы с целыми и дробными числами. Были достигнуты результаты:

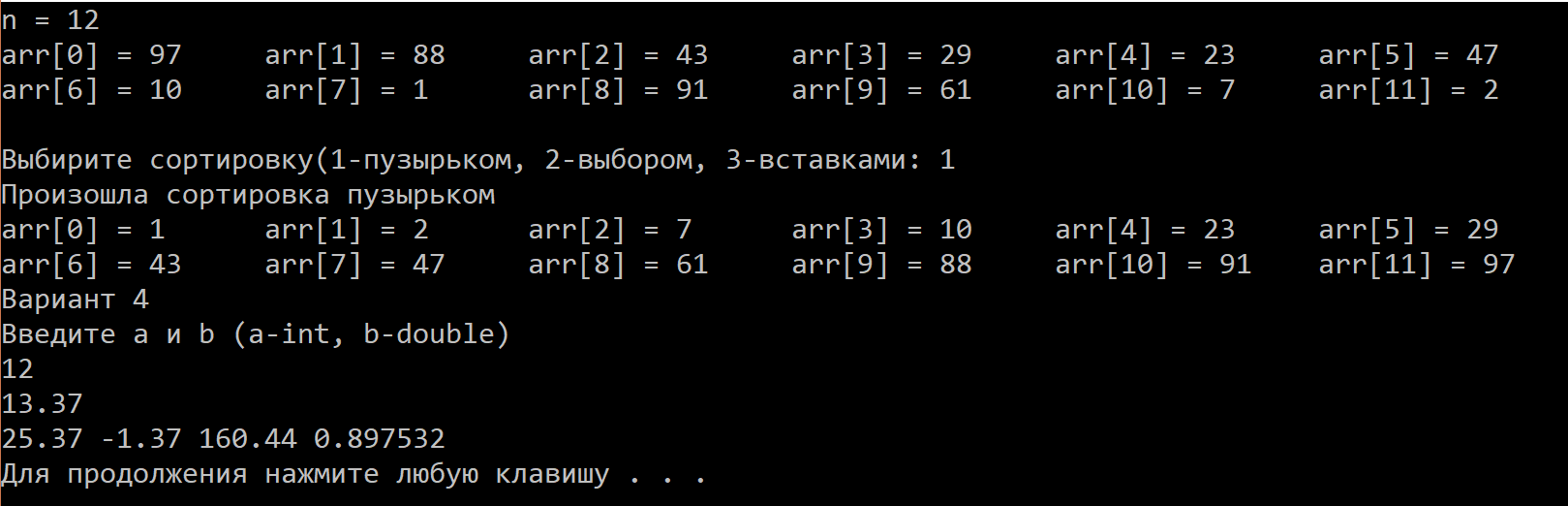


Рис. 11

<https://github.com/FunFuhrer/OOP/blob/master/practika5.cpp>

# Практическая работа №6

**Задание:**

1. Написать иерархию классов, описывающих имущество

налогоплательщиков. Она должна состоять из абстракт- ного

базового класса Property и производных от него классов Appartment, Car и CountryHouse. Базовый класс должен иметь поле worth(стоимость), конструктор с одним параметром, заполняющий

это поле, и чисто виртуальный метод рассчета налога,

переопределенный в каждом из производных классов. Налог на

квартиру вычисляется как 1/1000 ее стоимости, на машину — 1/200,

на дачу — 1/500. Также, каждый производный класс должен иметь

конструктор с одним параметром, передающий свой параметр

конструктору базового класса. В функции main завести массив из 7

указателей на Property и заполнить его указателями на

динамические объекты производных классов (первые 3 —

Appartment, следующие 2 — Car и последние 2 — CountryHouse).

Вывести на экран величину налога для всех 7 объектов. Не забудьте

также уничтожить динамические объекты перед завершением

программы.

1. Написать набор классов, представляющий выражения. В этом

наборе должен быть один абстрактный базовый тип, а также набор

производных от него типов по видам выражений (константа,

переменная, сумма, разность, произведение, частное, sin, cos, exp,

ln). У каждого из классов должны быть следующие виртуальные

функции: напечатать выражение (без параметров), вычислить

выражение (параметр — значение переменной, результат —

значение выражения), вернуть производную выражения (без

параметров), создать копию выражения (тоже без параметров).

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Property { |
|  | protected: |
|  | float worth; |
|  | public: |
|  | Property(float worth) { |
|  | this->worth = worth; |
|  | } |
|  | virtual float raschet() =0; |
|  | ~Property() { |
|  | } |
|  | }; |
|  | class Appartment :public Property { |
|  | string name; |
|  | public: |
|  | Appartment(string name, float worth) :Property(worth) { |
|  | this->name = name; |
|  | this->worth = worth; |
|  | } |
|  | float raschet() override { |
|  | return worth / 1000; |
|  | } |
|  | }; |
|  | class CountryHouse :public Property { |
|  | float square; |
|  | public: |
|  | CountryHouse(float square, float worth) :Property(worth) { |
|  | this->square = square; |
|  | this->worth = worth; |
|  | } |
|  | float raschet() override { |
|  | return worth / 500; |
|  | } |
|  | }; |
|  |  |
|  | class Car :public Property { |
|  | string brand; |
|  | public: |
|  | Car(string brand, float worth) :Property(worth) { |
|  | this->brand = brand; |
|  | this->worth = worth; |
|  | } |
|  | float raschet() override { |
|  | return worth / 200; |
|  | } |
|  | }; |
|  |  |
|  | int main() { |
|  | Property\* arr[7]; |
|  | arr[0] = new Appartment("main",15000000); |
|  | arr[1] = new Appartment("rent room 1",3000000); |
|  | arr[2] = new Appartment("rent room 2",2500000); |
|  | arr[3] = new Car("Nissan",600000); |
|  | arr[4] = new Car("Skoda",1100000); |
|  | arr[5] = new CountryHouse(45,450000); |
|  | arr[6] = new CountryHouse(30,300000); |
|  | float cost = 0; |
|  | for (int i = 0; i < 7; i++) { |
|  | cost = cost + arr[i]->raschet(); |
|  | } |
|  | cout << "velichina naloga dlja vseh ob#ektov : " << cost; |
|  |  |
|  | system("pause"); |
|  | delete[] arr; |
|  | return 0; |
|  | } |

Вариант 2

|  |
| --- |
| #include <iostream> |
|  | #include <cmath> |
|  | using namespace std; |
|  |  |
|  | class Function { |
|  | public: |
|  | virtual void PrintVyrazhenie() = 0; |
|  | virtual double Vychislenie(double x) = 0; |
|  | virtual void BackProizvodnuyu() = 0; |
|  | virtual Function\* Copy() = 0; |
|  | }; |
|  | class Const : public Function { |
|  | float a; |
|  | public: |
|  | Const(float a) { |
|  | this->a = a; |
|  | }; |
|  | void PrintVyrazhenie() override { |
|  | cout << "f(x)=" << a << endl; |
|  | } |
|  | double Vychislenie(double x) override { |
|  | return a; |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=0" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Const c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Peremennaya : public Function { |
|  | float k,b; |
|  | public: |
|  | Peremennaya(float k, float b) { |
|  | this->k = k; |
|  | this->b = b; |
|  | }; |
|  | void PrintVyrazhenie() override { |
|  | if (b!=0) |
|  | cout << "f(x)=" << k <<"x"<<b<< endl; |
|  | else |
|  | cout << "f(x)=" << k << "x" << endl; |
|  | } |
|  | double Vychislenie(double x) override { |
|  | return k\*x+b; |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=" <<k<< endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Peremennaya c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Summa : public Function { |
|  | float u, v; |
|  | public: |
|  | Summa(float u, float v) { |
|  | this->u = u; |
|  | this->v = v; |
|  | }; |
|  | void PrintVyrazhenie() override { |
|  | cout << "f(x)=" << u << "+" << v << endl; |
|  | } |
|  | double Vychislenie(double x) override { |
|  | return u+v; |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=0" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Summa c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Raznost : public Function { |
|  | float u, v; |
|  | public: |
|  | Raznost(float u, float v) { |
|  | this->u = u; |
|  | this->v = v; |
|  | }; |
|  | void PrintVyrazhenie() override { |
|  | cout << "f(x)=" << u << "-" << v << endl; |
|  | } |
|  | double Vychislenie(double x) override { |
|  | return u - v; |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=0" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Raznost c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Proizvedenie : public Function { |
|  | float u, v; |
|  | public: |
|  | Proizvedenie(float u, float v) { |
|  | this->u = u; |
|  | this->v = v; |
|  | }; |
|  | void PrintVyrazhenie() override { |
|  | cout << "f(x)=" << u << "\*" << v << endl; |
|  | } |
|  | double Vychislenie(double x) override { |
|  | return u \* v; |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=0" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Proizvedenie c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Chastnoe : public Function { |
|  | float u, v; |
|  | public: |
|  | Chastnoe(float u, float v) { |
|  | this->u = u; |
|  | this->v = v; |
|  | }; |
|  | void PrintVyrazhenie() override { |
|  | cout << "f(x)=" << u << "/" << v << endl; |
|  | } |
|  | double Vychislenie(double x) override { |
|  | if (v != 0) |
|  | return u / v; |
|  | else |
|  | return INFINITY; |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=0" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Chastnoe c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Sin : public Function { |
|  | float x; |
|  | public: |
|  | Sin(float x) { |
|  | this->x = x; |
|  | }; |
|  | void PrintVyrazhenie() override { |
|  | cout << "f(x)=sin(x)" << endl; |
|  | } |
|  | double Vychislenie(double x) override { |
|  | return sin(x); |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=cos(x)" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Sin c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Cos : public Function { |
|  | float x; |
|  | public: |
|  | Cos(float x) { |
|  | this->x = x; |
|  | }; |
|  | void PrintVyrazhenie() override { |
|  | cout << "f(x)=cos(x)" << endl; |
|  | } |
|  | double Vychislenie(double x) override { |
|  | return cos(x); |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=-sin(x)" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Cos c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Exp : public Function { |
|  | float x; |
|  | public: |
|  | Exp(float x) { |
|  | this->x = x; |
|  | }; |
|  | void PrintVyrazhenie()override { |
|  | cout << "f(x)=e^x" << endl; |
|  | } |
|  | double Vychislenie(double x)override { |
|  | return exp(x); |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=e^x" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Exp c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | class Ln : public Function { |
|  | float x; |
|  | public: |
|  | Ln(float x) { |
|  | this->x = x; |
|  | }; |
|  | void PrintVyrazhenie()override { |
|  | cout << "f(x)=ln(x)" << endl; |
|  | } |
|  | double Vychislenie(double x)override { |
|  | return log(x); |
|  | } |
|  | void BackProizvodnuyu()override { |
|  | cout << "f'(x)=1/x" << endl; |
|  | } |
|  | Function\* Copy()override { |
|  | Ln c = \*this; |
|  | return &c; |
|  | } |
|  | }; |
|  |  |
|  | int main() { |
|  | double y, x,k,b; |
|  |  |
|  | cout << "Const" << endl << "Enter x: "; |
|  | cin >> x; |
|  | Const a(x); |
|  | a.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << a.Vychislenie(x) << endl; |
|  | a.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Peremennaya" << endl << "k : "; |
|  | cin >> k; |
|  | cout << endl << "b : "; |
|  | cin >> b; |
|  | Peremennaya d(k, b); |
|  | cout << "Enter argument: "; |
|  | cin >> x; |
|  | d.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << d.Vychislenie(x) << endl; |
|  | d.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Sum" << endl << "Enter u and v: "; |
|  | cin >> y >> x; |
|  | Summa c(x,y); |
|  | c.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << c.Vychislenie(x) << endl; |
|  | c.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Raznost" << endl << "Enter u and v: "; |
|  | cin >> x >> y; |
|  | Raznost e(x,y); |
|  | e.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << e.Vychislenie(x) << endl; |
|  | e.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Proizvedenie" << endl << "Enter u and v: "; |
|  | cin >> x >> y; |
|  | Proizvedenie f(x, y); |
|  | f.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << f.Vychislenie(x) << endl; |
|  | f.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Chastnoe" << endl << "Enter u and v: "; |
|  | cin >> x >> y; |
|  | Chastnoe g(x, y); |
|  | g.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << g.Vychislenie(x) << endl; |
|  | g.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Sin" << endl << "Enter x : "; |
|  | cin >> x ; |
|  | Sin h(x); |
|  | h.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << h.Vychislenie(x) << endl; |
|  | h.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Cos" << endl << "Enter x : "; |
|  | cin >> x ; |
|  | Cos z(x); |
|  | z.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << z.Vychislenie(x) << endl; |
|  | z.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Exp" << endl << "Enter x : "; |
|  | cin >> x ; |
|  | Exp l(x); |
|  | l.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << l.Vychislenie(x) << endl; |
|  | l.BackProizvodnuyu(); |
|  |  |
|  | cout << endl << "Ln" << endl << "Enter x : "; |
|  | cin >> x ; |
|  | Ln m(x); |
|  | m.PrintVyrazhenie(); |
|  | cout << "f(" << x << ") = " << m.Vychislenie(x) << endl; |
|  | m.BackProizvodnuyu(); |
|  |  |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Практической работы №6 были использованы библиотека iostream. Были реализованы методы, возвращающие налоги на собственность и методы для работы с классами математических выражений. Были достигнуты результаты:

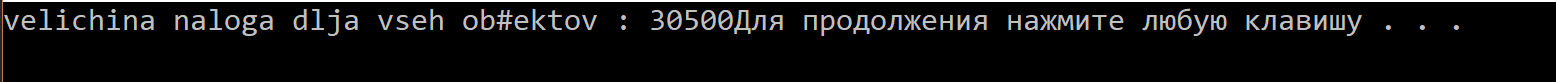


Рис. 12

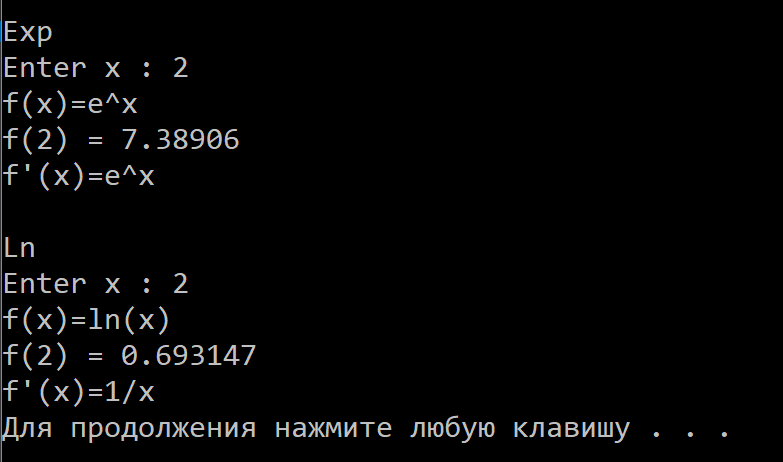
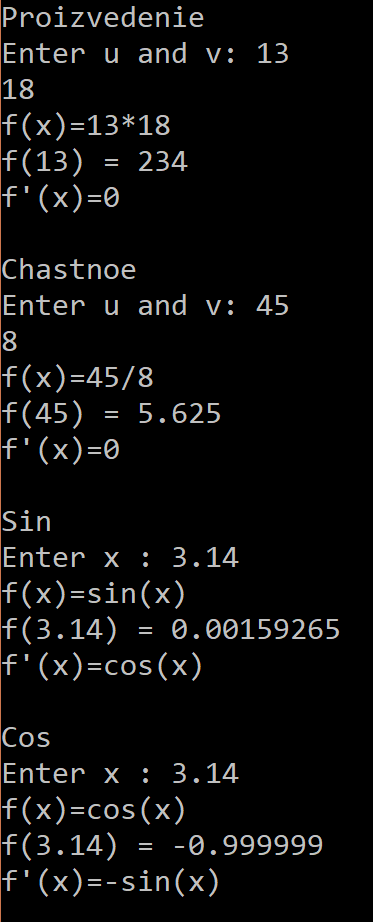
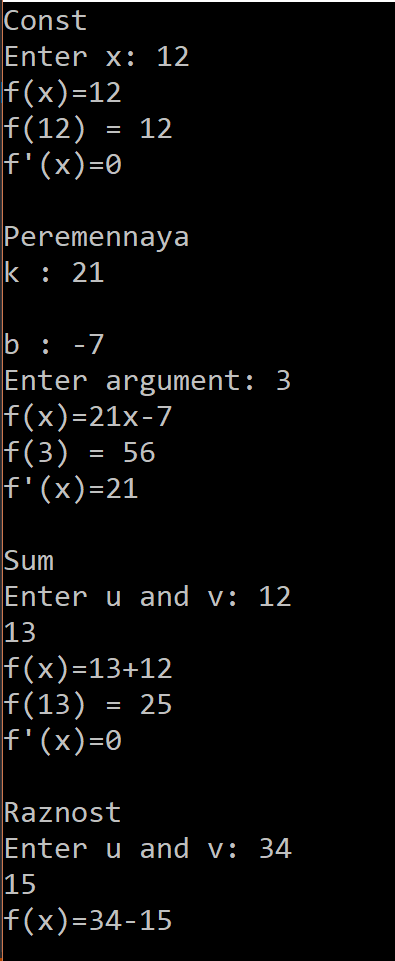


Рис. 13

<https://github.com/FunFuhrer/OOP/blob/master/practika6.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika6.2.cpp>

# Практическая работа №7

**Задание:**

1. Реализовать абстрактный класс «Животное» и путём

наследования от него получить классы «Кошка», «Собака»,«Попугай».

1. Реализовать абстрактный класс «Фигура» и путём наследования

от него получить абстрактный класс «Четырёхугольник», и затем

путём наследования получить классы «Ромб», «Прямоугольник».

1. Реализовать абстрактный класс «Транспортное средство» и

путём наследования от него получить классы «Автомобиль», «Автобус», «Велосипед».

1. Реализовать абстрактные классы «Экран» и «Клавиатура», путём

наследования от них получить классы «Ноутбук», «Телефон»,

«Стационарный компьютер».

**Ход выполнения:**

Вариант 1

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Animal { |
|  | public: |
|  | virtual void MakeVoice() = 0; |
|  | }; |
|  | class Cat : public Animal { |
|  | public: |
|  | void MakeVoice() override { |
|  | cout << "Meow!!!" << endl; |
|  | } |
|  | }; |
|  | class Dog : public Animal { |
|  | public: |
|  | void MakeVoice() override { |
|  | cout << "Woof!!!" << endl; |
|  | } |
|  | }; |
|  | class Parrot : public Animal { |
|  | public: |
|  | void MakeVoice() override { |
|  | cout << "WTF!!!" << endl; |
|  | } |
|  | }; |
|  | int main() { |
|  | Cat a; |
|  | Dog b; |
|  | Parrot c; |
|  | Animal \*voice1 = &a; |
|  | voice1->MakeVoice(); |
|  | Animal \*voice2 = &b; |
|  | voice2->MakeVoice(); |
|  | Animal \*voice3 = &c; |
|  | voice3->MakeVoice(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 2

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Figure{ |
|  | public: |
|  | virtual void Square() = 0; |
|  | }; |
|  | class Quadrilateral : public Figure { |
|  | public: |
|  | virtual void Square() = 0; |
|  | }; |
|  | class Rhombus : public Quadrilateral { |
|  | double d1, d2; |
|  | public: |
|  | Rhombus(double d1, double d2) { |
|  | this->d1 = d1; |
|  | this->d2 = d2; |
|  | } |
|  | void Square() override { |
|  | cout<< d1\*d2 / 2<<endl; |
|  | } |
|  | }; |
|  | class Rectangle : public Quadrilateral { |
|  | double a, b; |
|  | public: |
|  | Rectangle(double a, double b) { |
|  | this->a = a; |
|  | this->b = b; |
|  | } |
|  | void Square() override { |
|  | cout<< a \* b<<endl; |
|  | } |
|  | }; |
|  | int main() { |
|  | Rectangle a(13, 37); |
|  | Rhombus b(13, 37); |
|  | Figure \*square1 = &a; |
|  | square1->Square(); |
|  | Figure \*square2 = &b; |
|  | square2->Square(); |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 3

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Vehicle { |
|  | public: |
|  | virtual void CountPassenger() = 0; |
|  | }; |
|  | class Auto : public Vehicle { |
|  | public: |
|  | void CountPassenger() override { |
|  | cout << "5 Passenger" << endl; |
|  | } |
|  | }; |
|  | class Bus : public Vehicle { |
|  | public: |
|  | void CountPassenger() override { |
|  | cout << "1000000 Passenger" << endl; |
|  | } |
|  | }; |
|  | class Bicycle : public Vehicle { |
|  | public: |
|  | void CountPassenger() override { |
|  | cout <<"1 Passenger" << endl; |
|  | } |
|  | }; |
|  | int main() { |
|  | Bicycle a; |
|  | Auto b; |
|  | Bus c; |
|  | Vehicle \*passenger1 = &a; |
|  | passenger1->CountPassenger(); |
|  | Vehicle \*passenger2 = &b; |
|  | passenger2->CountPassenger(); |
|  | Vehicle \*passenger3 = &c; |
|  | passenger3->CountPassenger(); |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

Вариант 4

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | class Screen { |
|  | protected: |
|  | double width, length; |
|  | public: |
|  | virtual void diagonal() = 0; |
|  | }; |
|  | class Keyboard { |
|  | protected: |
|  | int key; |
|  | public: |
|  | virtual void countkey() = 0; |
|  | }; |
|  | class Notebook :public Screen, public Keyboard { |
|  | string brand; |
|  | public: |
|  | void setBrand(string brand) { |
|  | this->brand = brand; |
|  | } |
|  | void printBrand() { |
|  | cout << "Brand : " <<brand<< endl; |
|  | } |
|  | void diagonal() override { |
|  | cout << "Diagonali = 17.3" << endl; |
|  | } |
|  | void countkey() override { |
|  | cout << "countkey = 50" << endl; |
|  | } |
|  | }; |
|  | class Phone :public Screen, public Keyboard { |
|  | string brand; |
|  | public: |
|  | void setBrand(string brand) { |
|  | this->brand = brand; |
|  | } |
|  | void printBrand() { |
|  | cout << "Brand : " << brand << endl; |
|  | } |
|  | void diagonal() override { |
|  | cout << "Diagonali = 6.7" << endl; |
|  | } |
|  | void countkey() override { |
|  | cout << "countkey = 0" << endl; |
|  | } |
|  | }; |
|  | class PC :public Screen, public Keyboard { |
|  | string brand; |
|  | public: |
|  | void setBrand(string brand) { |
|  | this->brand = brand; |
|  | } |
|  | void printBrand() { |
|  | cout << "Brand : " << brand << endl; |
|  | } |
|  | void diagonal() override { |
|  | cout << "Diagonali = 35" << endl; |
|  | } |
|  | void countkey() override { |
|  | cout << "countkey = 70" << endl; |
|  | } |
|  | }; |
|  |  |
|  | int main() { |
|  | Notebook a; |
|  | a.setBrand("MSI"); |
|  | Phone b; |
|  | b.setBrand("SONY"); |
|  | PC c; |
|  | c.setBrand("Noname"); |
|  | Screen \*data1 = &a; |
|  | Keyboard \*data2 = &a; |
|  | a.printBrand(); |
|  | data1->diagonal(); |
|  | data2->countkey(); |
|  | cout << endl; |
|  |  |
|  | Screen \*data3 = &b; |
|  | Keyboard \*data4 = &b; |
|  | b.printBrand(); |
|  | data3->diagonal(); |
|  | data4->countkey(); |
|  | cout << endl; |
|  |  |
|  | Screen \*data5 = &c; |
|  | Keyboard \*data6 = &c; |
|  | c.printBrand(); |
|  | data5->diagonal(); |
|  | data6->countkey(); |
|  | cout << endl; |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Практической работы №7 были использованы библиотеки iostream, string. Была реализованы методы для демонстрации работоспособности производных классов. Были достигнуты результаты:

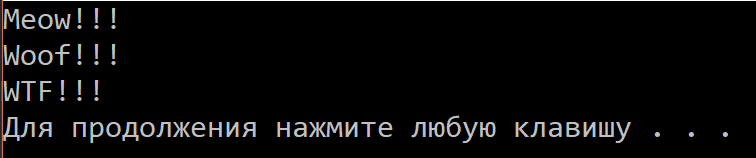


Рис. 14

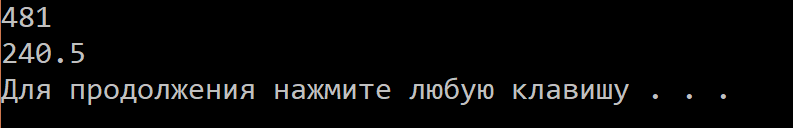


Рис. 15

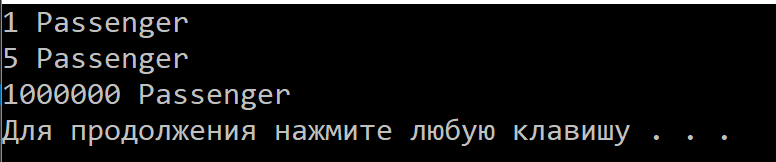


Рис. 16

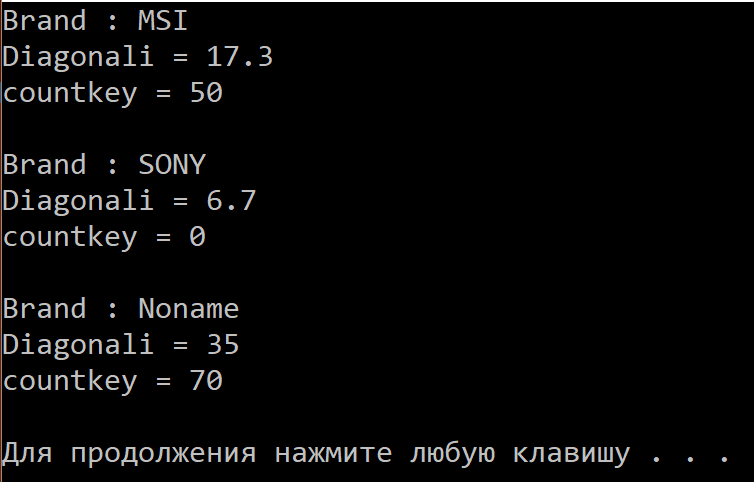


Рис. 17

<https://github.com/FunFuhrer/OOP/blob/master/practika7.1.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika7.2.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika7.3.cpp>

<https://github.com/FunFuhrer/OOP/blob/master/practika7.4.cpp>

# Практическая работа №8

**Задание:**

1. Реализуйте программу, считывающую текст из файла и выводящую

каждое слово с новой строки.

1. Реализуйте программу, считывающую текст с клавиатуры и

записывающую его в файл.

**Ход выполнения:**

|  |
| --- |
| #include <iostream> |
|  | #include <fstream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | int main() { |
|  | ofstream fout("a.txt"); |
|  | char c = 0; |
|  |  |
|  | while (c != '\n') { |
|  | cin.get(c); |
|  | fout << c; |
|  | } |
|  |  |
|  | fout.close(); |
|  |  |
|  | ifstream fin("a.txt"); |
|  |  |
|  | while (!fin.eof()) { |
|  | string s; |
|  | fin >> s; |
|  | cout << s << endl; |
|  | } |
|  |  |
|  | fin.close(); |
|  |  |
|  | system("pause"); |
|  | return 0; |
|  | } |

**Выводы:**

В ходе выполнения Практической работы №8 были использованы библиотеки iostream,string,fstream. Были достигнуты результаты:

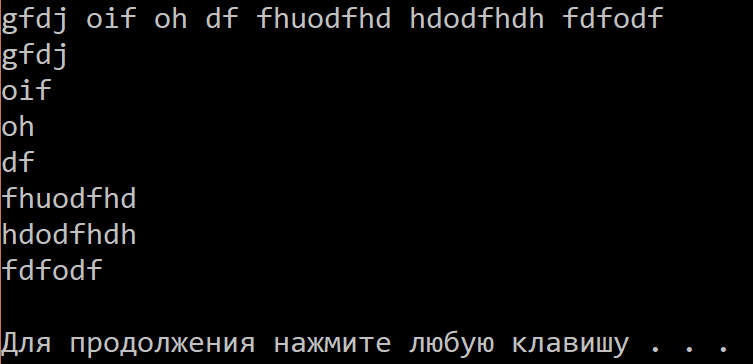


Рис. 18

<https://github.com/FunFuhrer/OOP/blob/master/practika8.cpp>