|  |  |
| --- | --- |
| 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 | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ** | |
| **по дисциплине** | |
| **«**Объектно-ориентированное программирование**»** | |
|  | |
| Выполнил студент группы ИНБО-04-18 | Автушко Д.А. |
| Принял ассистент кафедры | Хлебникова В.Л. |

|  |  |  |
| --- | --- | --- |
| Лабораторные работы выполнены | «23» февраля 2019 г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. |  |

Москва 2019

## Таблица успеваемости

**Практические занятия**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №/р. | страница в отчете. | Студент выполнил:  (подпись) | Преподаватель принял:  (подпись) | Балл  (максимально возможный) | Балл (Фактический) |
| 1. |  |  |  | 2 | … |
| 2. |  |  |  | 2 |  |
| 3. |  |  |  | 4 |  |
| 4. |  |  |  | 6 |  |
| 5. |  |  |  | 6 |  |
| 6. |  |  |  | 8 |  |
| 7. |  |  |  | 8 |  |
|  |  |  | Итоговый балл: | 36 |  |

## Практическая работа №1

**Изучение синтаксиса языка C++**

**Цель практической работы**

Целью данной практической работы является знакомство с синтаксисом, основными алгоритмическими конструкциями языка С++, а также приобретение практических навыков разработки объектно-ориентированных программ на языке программирования С++.

**Задачи**

1. Реализовать программу, которая считывает имя и здоровается
2. Реализовать программу, производящую операции над числами. (Сложение/вычитание, целые числа/числа с плавающей запятой)

**Ход работы**

1. Реализовать программу, которая считывает имя и здоровается

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  | int main(int argc, const char \* argv[]) { |
|  | string name; |
|  | cout << "Enter your name: "; |
|  | cin >> name; |
|  | cout << "Hello, " << name << endl; |
|  | return 0; |
|  | } |

1. Реализовать программу, производящую операции над числами. (Сложение/вычитание, целые числа/числа с плавающей запятой)

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  | int main(int argc, const char \* argv[]) { |
|  | int a,b; |
|  | cin >> a >> b; |
|  | cout << "a + b = " << a+b << "\n" << "a - b = " << a-b << endl; |
|  | double c,d; |
|  | cin >> c >> d; |
|  | cout << "c + d = " << c+d << "\n" << "c - d = " << c-d << endl; |
|  | return 0; |
|  | } |

**Вывод**

В ходе выполнения практической работы была использована библиотека стандартного ввода и вывода, операторы ввода, вывода, сложения и вычитания. Были достигнуты результаты, изображённые на скриншотах (рис. 1 и 2). Также исходный код программ был помещён на GitHub и доступен по ссылкам:

https://github.com/adanil/OOP/tree/master/Practica

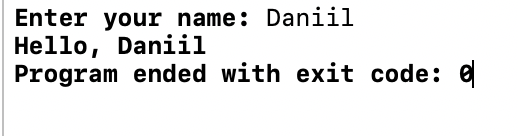


Рис. 1 Результат работы первой программы

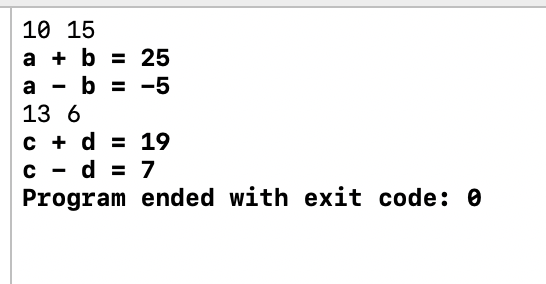


Рис. 2 Результат работы второй программы

## Практическая работа №2

**Решение задач на нелинейные алгоритмические конструкции языка С++**

**Цель практической работы**

Целью данной практической работы является получение практических навыков по разработке нелинейных алгоритмов с использованием конструкции «if-else».

**Ход работы**

|  |
| --- |
| #include <iostream> |
|  | #include <cmath> |
|  |  |
|  | using namespace std; |
|  | int main(int argc, const char \* argv[]) { |
|  | double a = 4.8, b = -7.9; |
|  | double n,m; |
|  | if (a >= b) |
|  | n = cbrt(a-b); |
|  | else |
|  | n = a \* a + (a-b)/sin(a\*b); |
|  | if (n < b) |
|  | m = (n+a)/-b + sqrt(sin(a)\*sin(a) - cos(n)); |
|  | else if (n == b) |
|  | m = b\*b + tan(n\*a); |
|  | else |
|  | m = b\*b\*b + n\*a\*a; |
|  | cout << "a = " << a << " b = " << b << " m = " << m << " n = " << n << endl; |
|  |  |
|  | //Вариант 2 |
|  | double a2 = -0.6,b2 = 5.3; |
|  | double z,t; |
|  | if (a < b) |
|  | z = sqrt(abs(a2\*a2 - b2\*b2)); |
|  | else |
|  | z = 1 - 2\*cos(a2)\*sin(b2); |
|  | if (z < b2) |
|  | t = cbrt(z+a2\*a2\*b2); |
|  | else if (z == b2) |
|  | t = 1 - log10(z) + cos(a2\*a2\*b2); |
|  | else |
|  | t = 1/cos(z\*a2); |
|  | cout << "a = " << a2 << " b = " << b2 << " z = " << z << " t = " << t << endl; |
|  |  |
|  | //Вариант 3 |
|  | double a3 = 4.6,b3 = 1.5; |
|  | double y3,t3; |
|  | if (a3 <= b3) |
|  | y3 = ((a3 - b3)/(a3 + b3)) \* ((a3+b3)/(a3\*a3 - a3\*b3 + b3\*b3)); |
|  | else |
|  | y3 = a3 + log(b3\*b3); |
|  | if (y3 == b3) |
|  | t3 = (2\*y3 + sqrt(y3\*y3 - a3))/(2\*b3 - sqrt(a3\*a3 - y3)); |
|  | else if (y3 < b3) |
|  | t3 = sin(y3)\*sin(y3) + 1/(tan(a3-b3)); |
|  | else |
|  | t3 = cbrt(y3\*sin(a3)) + 1/(sqrt(y3\*cos(b3))); |
|  | cout << "a = " << a3 << " b = " << b3 << " y = " << y3 << " t = " << t3 << endl; |
|  |  |
|  | //Вариант 4 |
|  | double a4 = 3.5,x4 = -0.7; |
|  | double y4,t4; |
|  | if (a4 <= x4) |
|  | y4 = a4 + log(x4+a4); |
|  | else |
|  | y4 = sqrt(sin(a4\*x4)); |
|  | if (a4 > y4) |
|  | t4 = y4/(a4-x4); |
|  | else if (a4 == y4) |
|  | t4 = (y4/(a4-x4)) + ((a4+x4)/(y4\*y4)); |
|  | else |
|  | t4 = tan(a4\*x4) + cos(2\*a4\*y4); |
|  | cout << "a = " << a4 << " x = " << x4 << " y = " << y4 << " t = " << t4 << endl; |
|  |  |
|  |  |
|  | return 0; |
|  | } |

**Вывод**

В ходе выполнения практической работы была использована библиотека cmath, которая позволила вычислять сложные математические функции, а также быстро совершить проверку на ошибки, и конструкция ветвления «if-else». Были достигнуты результаты, изображённые на скриншотах (рис. 3-6). Также код программ доступен по ссылкам:

https://github.com/adanil/OOP/tree/master/Practica

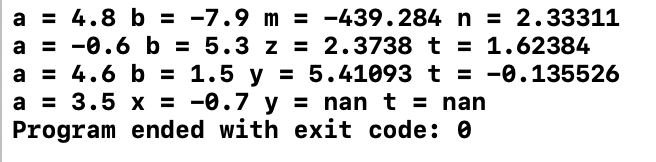


Рис. 3 Результат работы программы

## Практическая работа №3

**Классы в языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков при разработке классов на языке программирования С++.

**Задачи**

1. Реализовать класс «Книга»
2. Реализовать класс «Группа студентов»
3. Реализовать класс «Автомобиль»
4. Реализовать класс «Вектор»
5. Реализовать класс «Библиотека»

**Ход работы**

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include <cmath> |
|  | #include <vector> |
|  | using namespace std; |
|  |  |
|  | class Book{ |
|  | string name,author; |
|  | int pages,yearOfIssue; |
|  | public: |
|  | Book(string nam,string aut): name(nam),author(aut){} |
|  | string getName(){ |
|  | return name; |
|  | } |
|  | string getAuthor(){ |
|  | return author; |
|  | } |
|  | void setName(string n){ |
|  | this->name = n; |
|  | } |
|  | void setAuthor(string a){ |
|  | this->author = a; |
|  | } |
|  | int getPages(){ |
|  | return pages; |
|  | } |
|  | int getYear(){ |
|  | return yearOfIssue; |
|  | } |
|  | void setPages(int p){ |
|  | this->pages = p; |
|  | } |
|  | void setYear(int y){ |
|  | this->yearOfIssue = y; |
|  | } |
|  | }; |
|  | class StudentsGroup{ |
|  | string nameOfGroup; |
|  | int countOfMembers,countExams; |
|  | public: |
|  | void setName(string n){ |
|  | nameOfGroup = n; |
|  | } |
|  | void setMembers(int m){ |
|  | countOfMembers = m; |
|  | } |
|  | void setExams(int e){ |
|  | countExams = e; |
|  | } |
|  | string getName(){ |
|  | return nameOfGroup; |
|  | } |
|  | int getMembers(){ |
|  | return countOfMembers; |
|  | } |
|  | int getExams(){ |
|  | return countExams; |
|  | } |
|  |  |
|  | }; |
|  | class Auto{ |
|  | string brand,model; |
|  | int year; |
|  | public: |
|  | void setBrand(string n){ |
|  | brand = n; |
|  | } |
|  | void setModel(int m){ |
|  | model = m; |
|  | } |
|  | void setYear(int e){ |
|  | year = e; |
|  | } |
|  | string getBrand(){ |
|  | return brand; |
|  | } |
|  | string getModel(){ |
|  | return model; |
|  | } |
|  | int getYear(){ |
|  | return year; |
|  | } |
|  | }; |
|  | class Vector{ |
|  | int module; |
|  | double x1,x2,y1,y2; |
|  | public: |
|  | void setStartPoint(double x,double y){ |
|  | x1 = x; |
|  | y1 = y; |
|  | } |
|  | void setEndPoint(double x,double y){ |
|  | x2 = x; |
|  | y2 = y; |
|  | } |
|  |  |
|  | double getModule(){ |
|  | return sqrt((x1 - x2)\*(x1 - x2) + (y1 - y2)\*(y1 - y2)); |
|  | } |
|  | }; |
|  | class Librar{ |
|  | string city,country,street; |
|  | int countBooks; |
|  | vector<Book> books; |
|  | public: |
|  | void setCity(string ct){ |
|  | city = ct; |
|  | } |
|  | void setCountry(string c){ |
|  | country = c; |
|  | } |
|  | void setStreet(string st){ |
|  | street = st; |
|  | } |
|  | void setCountBooks(int cnt){ |
|  | countBooks = cnt; |
|  | } |
|  | void addBook(Book a){ |
|  | books.push\_back(a); |
|  | } |
|  | string getCity(string ct){ |
|  | return city; |
|  | } |
|  | string getCountry(string c){ |
|  | return country; |
|  | } |
|  | string getStreet(string st){ |
|  | return street; |
|  | } |
|  | int getCountBooks(int cnt){ |
|  | return countBooks; |
|  | } |
|  | }; |
|  | int main(int argc, const char \* argv[]) { |
|  | Book myBook("Wind","Bethoven"); |
|  | cout << myBook.getName() << ' ' << myBook.getAuthor() << endl; |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. Для этого были использованы библиотека STL, перегрузка оператора равенства, шаблоны функций. Все задачи были выполнены в виде много файлового проекта с разбиением классов на файлы объявления и реализации класса. Результаты работы программ приведены в скриншотах (рис. 7-11). Код программ также доступен на GitHub по ссылкам:

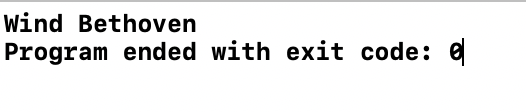
https://github.com/adanil/OOP/tree/master/Practica

Рис. 4 Результат работы программы

## Практическая работа №4

**Динамическое выделение памяти для объектов в языке С++**

**Цель практической работы**

Целью данной практической работы является изучение динамического выделения и удаления памяти, а так же работа с указателями на языке С++.

**Задачи**

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, четвёртая – освободить занимаемую память. Программа должна предлагать продолжать работу (создавать новые массивы) или выйти из программы.
2. Объявите указатель на массив типа int и выделите память для 12-ти элементов. Необходимо написать функцию, которая поменяет значения четных и нечетных ячеек массива.
3. 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. Количество строк и столбцов выбирает пользователь

**Ход работы**

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | #include <cstdlib> |
|  | using namespace std; |
|  |  |
|  | double\* Create(int n){ |
|  | double\* m = new double[n]; |
|  | return m; |
|  | } |
|  |  |
|  | void Init(double\* m,int n){ |
|  | for (int i = 0;i < n;i++) |
|  | m[i] = i; |
|  | } |
|  |  |
|  | void Print(double\* m,int n){ |
|  | for (int i = 0;i < n;i++) |
|  | cout << m[i] << ' '; |
|  | cout << endl; |
|  | } |
|  |  |
|  | void MemFree(double\*m){ |
|  | delete[] m; |
|  | } |
|  |  |
|  | void InitRandom(int\*\* a,int k,int l){ |
|  | for (int i = 0;i < k;i++) |
|  | for (int j = 0;j < l;j++) |
|  | a[i][j] = (rand()%40) + 10; |
|  | } |
|  |  |
|  | void PrintIntArray(int\*\* a,int k,int l){ |
|  | for (int i = 0;i < k;i++){ |
|  | for (int j = 0;j < l;j++) |
|  | cout << a[i][j] << ' '; |
|  | cout << endl; |
|  | } |
|  | } |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | int n = 0; |
|  | while (n != -1){ |
|  | cout << "Для выхода из программы введите -1" << endl; |
|  | cout << "Введите размер массива: "; |
|  | cin >> n; |
|  | if (n == -1) |
|  | break; |
|  | double\* m = Create(n); |
|  | Init(m, n); |
|  | Print(m, n); |
|  | MemFree(m); |
|  | } |
|  | //Вариант 2 |
|  | int\* arr = new int[12]; |
|  | for (int i = 0;i < 12;i++){ |
|  | arr[i] = i; |
|  | cout << arr[i] << ' '; |
|  | } |
|  | cout << endl; |
|  | for (int i = 0;i < 11;i += 2){ |
|  | int t = arr[i]; |
|  | arr[i] = arr[i+1]; |
|  | arr[i+1] = t; |
|  | } |
|  | for (int i = 0;i < 12;i++) |
|  | cout << arr[i] << ' '; |
|  | cout << endl; |
|  |  |
|  | //Вариант 3 |
|  | int k,l; |
|  | cout << "Введите количество строк в массиве: "; |
|  | cin >> k; |
|  | cout << "Введите количество столбов в массиве: "; |
|  | cin >> l; |
|  | int\*\* a = new int\*[k]; |
|  | for (int i = 0; i < k;i++) |
|  | a[i] = new int[l]; |
|  | InitRandom(a, k, l); |
|  | PrintIntArray(a, k, l); |
|  |  |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. При выполнении использовались массивы с динамическим выделением памяти, которые создавались с помощью генератора случайных чисел. Для генератора случайных чисел была подключена библиотека ctime. Результат работы программ приведён на скриншотах (рис.12-14). Так же исходный код программ доступен на GitHub по ссылкам:

https://github.com/adanil/OOP/tree/master/Practica

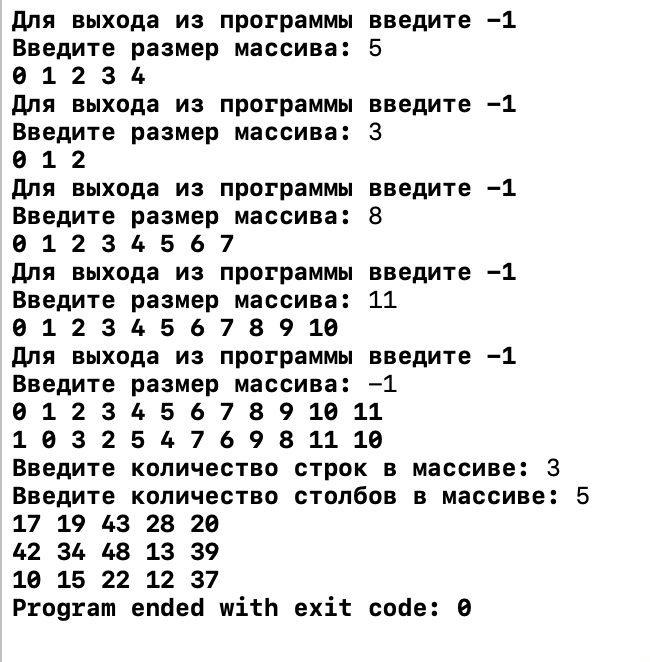


Рис. 5 Результат работы программы

## Практическая работа №5

**Перегрузка функций**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков по программированию перегрузки функций на языке C++.

**Задачи**

1. Реализовать сортировку пузырьком для целых чисел, а затем перегрузить её для дробных.
2. Реализовать сортировку выбором для целых чисел, а затем перегрузить её для дробных.
3. Реализовать сортировку вставками для целых чисел, а затем перегрузить её для дробных.
4. Реализовать программу-калькулятор, работающую с разными типами данных.

**Ход работы**

|  |
| --- |
| #include <iostream> |
|  |  |
|  | using namespace std; |
|  |  |
|  | void BubbleSort(int\* arr,int sizeArr){ |
|  | for (int i = 0;i < sizeArr-1;i++) |
|  | for (int j = i+1;j < sizeArr;j++) |
|  | if (arr[i] > arr[j]){ |
|  | int t = arr[i]; |
|  | arr[i] = arr[j]; |
|  | arr[j] = t; |
|  | } |
|  | } |
|  | void BubbleSort(double\* arr,int sizeArr){ |
|  | for (int i = 0;i < sizeArr-1;i++) |
|  | for (int j = i+1;j < sizeArr;j++) |
|  | if (arr[i] > arr[j]){ |
|  | double t = arr[i]; |
|  | arr[i] = arr[j]; |
|  | arr[j] = t; |
|  | } |
|  | } |
|  |  |
|  | void SelectionSort(int\* arr,int sizeArr){ |
|  | for (int i = 0;i < sizeArr-1;i++){ |
|  | int minEl = arr[i]; |
|  | int minIn = i; |
|  | for (int j = i;j < sizeArr;j++) |
|  | if (arr[j] < minEl){ |
|  | minEl = arr[j]; |
|  | minIn = j; |
|  | } |
|  | arr[minIn] = arr[i]; |
|  | arr[i] = minEl; |
|  | } |
|  | } |
|  |  |
|  | void SelectionSort(double\* arr,int sizeArr){ |
|  | for (int i = 0;i < sizeArr-1;i++){ |
|  | double minEl = arr[i]; |
|  | int minIn = i; |
|  | for (int j = i;j < sizeArr;j++) |
|  | if (arr[j] < minEl){ |
|  | minEl = arr[j]; |
|  | minIn = j; |
|  | } |
|  | arr[minIn] = arr[i]; |
|  | arr[i] = minEl; |
|  | } |
|  | } |
|  |  |
|  | void InsertionSort(int\* arr,int sizeArr){ |
|  | for (int i = 1;i < sizeArr;i++){ |
|  | int j = i; |
|  | while (arr[j] < arr[j-1] && j > 0) { |
|  | int t = arr[j-1]; |
|  | arr[j-1] = arr[j]; |
|  | arr[j] = t; |
|  | j--; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | int sum(int a,int b){ |
|  | return a+b; |
|  | } |
|  | double sum(double a,double b){ |
|  | return a+b; |
|  | } |
|  | float sum(float a,float b){ |
|  | return a+b; |
|  | } |
|  | long long sum(long long a,long long b){ |
|  | return a+b; |
|  | } |
|  | int diff(int a,int b){ |
|  | return a-b; |
|  | } |
|  | double diff(double a,double b){ |
|  | return a-b; |
|  | } |
|  | float diff(float a,float b){ |
|  | return a-b; |
|  | } |
|  | long long diff(long long a,long long b){ |
|  | return a-b; |
|  | } |
|  | int mul(int a,int b){ |
|  | return a\*b; |
|  | } |
|  | double mul(double a,double b){ |
|  | return a\*b; |
|  | } |
|  | float mul(float a,float b){ |
|  | return a\*b; |
|  | } |
|  | long long mul(long long a,long long b){ |
|  | return a\*b; |
|  | } |
|  | int divide(int a,int b){ |
|  | return a/b; |
|  | } |
|  | double divide(double a,double b){ |
|  | return a/b; |
|  | } |
|  | float divide(float a,float b){ |
|  | return a/b; |
|  | } |
|  | long long divide(long long a,long long b){ |
|  | return a/b; |
|  | } |
|  | void InsertionSort(double\* arr,int sizeArr){ |
|  | for (int i = 1;i < sizeArr;i++){ |
|  | int j = i; |
|  | while (arr[j] < arr[j-1] && j > 0) { |
|  | double t = arr[j-1]; |
|  | arr[j-1] = arr[j]; |
|  | arr[j] = t; |
|  | j--; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | int arr[10] = {5,7,23,43,1,23,7,8,19,31}; |
|  | for (int i = 0;i < 10;i++) |
|  | cout << arr[i] << ' '; |
|  | cout << endl; |
|  | //BubbleSort(arr, 10); |
|  | //SelectionSort(arr, 10); |
|  | InsertionSort(arr, 10); |
|  | for (int i = 0;i < 10;i++) |
|  | cout << arr[i] << ' '; |
|  | cout << endl; |
|  | double darr[10] = {1.3,1.5,13.2,18.9,2.7,6.0,15.3,3.4,8.3,10.1}; |
|  | for (int i = 0;i < 10;i++) |
|  | cout << darr[i] << ' '; |
|  | cout << endl; |
|  | //BubbleSort(darr, 10); |
|  | //SelectionSort(darr, 10); |
|  | InsertionSort(darr, 10); |
|  | for (int i = 0;i < 10;i++) |
|  | cout << darr[i] << ' '; |
|  | cout << endl; |
|  | //Вариант 4 |
|  | cout << "Задание 4" << endl; |
|  | //int a = 5; |
|  | //int b = 3; |
|  | double a = 7.7; |
|  | double b = 13.8; |
|  | cout << sum(a,b) << ' ' << diff(a,b) << ' ' << mul(a,b) << ' ' << divide(a,b) << endl; |
|  |  |
|  |  |
|  | return 0; |
|  | } |

**Вывод**

Все задачи были выполнены. В ходе выполнения была изучена техника перегрузки функций. Так же для заполнения массивов использовался генератор случайных чисел. Результат работы программы приведён на скриншотах (рис. 15-18). Исходный код программ так же доступен на GitHub по ссылкам:

https://github.com/adanil/OOP/tree/master/Practica

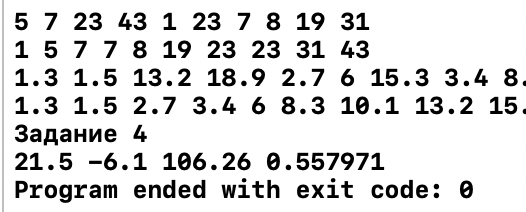


Рис. 6 Результат работы программы

## Практическая работа №6

**Решение задач на наследование на языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков использования принципа ООП – наследования для разработки программ на языке программирования C++.

**Задачи**

1. Написать иерархию классов, описывающих имущество налогоплательщиков. Она должна состоять из абстрактного 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 забудь также уничтожить динамические объекты перед завершением программы.
2. 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 работы**

|  |
| --- |
| #include <iostream> |
|  |  |
|  | using namespace std; |
|  |  |
|  | class Property{ |
|  | protected: |
|  | int worth; |
|  | public: |
|  | Property(int a):worth(a){} |
|  | virtual int tax() = 0; |
|  | }; |
|  |  |
|  | class Apartment:public Property{ |
|  | public: |
|  | int tax(){ |
|  | return worth/1000; |
|  | } |
|  | Apartment(int b = 0):Property(b){ |
|  | } |
|  | }; |
|  |  |
|  | class Car:public Property{ |
|  | public: |
|  | int tax(){ |
|  | return worth/200; |
|  | } |
|  | Car(int b = 0):Property(b){ |
|  | } |
|  | }; |
|  | class CountryHouse:public Property{ |
|  | public: |
|  | int tax(){ |
|  | return worth/500; |
|  | } |
|  | CountryHouse(int b = 0):Property(b){ |
|  | } |
|  | }; |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | Property\* arr[7]; |
|  | arr[0] = new Apartment(1000000);arr[1] = new Apartment(1500000);arr[2] = new Apartment(2000000); |
|  | arr[3] = new Car(300000);arr[4] = new Car(700000); |
|  | arr[5] = new CountryHouse(3000000);arr[6] = new CountryHouse(7000000); |
|  | for (int i = 0;i < 7;i++){ |
|  | cout << arr[i]->tax() << endl; |
|  | delete arr[i]; |
|  | } |
|  |  |
|  | return 0; |
|  | } |

Программа №2:

|  |
| --- |
| #include "zadanie2.hpp" |
|  | #include <iostream> |
|  | #include <cmath> |
|  |  |
|  | using namespace std; |
|  | ///Вариант 2 |
|  | class Expression{ |
|  | public: |
|  | virtual void print() = 0; |
|  | virtual int calc(int a) = 0; |
|  | virtual void derived() = 0; |
|  | virtual Expression\* copy() = 0; |
|  | }; |
|  |  |
|  | class Constant:Expression{ |
|  | int value; |
|  | public: |
|  | Constant(int a):value(a){} |
|  | void print(){ |
|  | cout << value << endl; |
|  | } |
|  | int calc(int a){ |
|  | return a; |
|  | } |
|  | void derived(){ |
|  | cout << "Constant derived is 0" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Constant a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Variable:Expression{ |
|  | int value; |
|  | public: |
|  | Variable(int a):value(a){} |
|  | void print(){ |
|  | cout << "x" << endl; |
|  | } |
|  | int calc(int a){ |
|  | return a; |
|  | } |
|  | void derived(){ |
|  | cout << "Variable derived is 1" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Variable a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Sum:Expression{ |
|  | int value; |
|  | public: |
|  | Sum(int a):value(a){} |
|  | void print(){ |
|  | cout << "a + b" << endl; |
|  | } |
|  | int calc(int a){ |
|  | return value + a; |
|  | } |
|  | void derived(){ |
|  | cout << "Sum derived is a' + b'" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Sum a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Difference:Expression{ |
|  | int value; |
|  | public: |
|  | Difference(int a):value(a){} |
|  | void print(){ |
|  | cout << "a - b" << endl; |
|  | } |
|  | int calc(int a){ |
|  | return value - a; |
|  | } |
|  | void derived(){ |
|  | cout << "Difference derived is a' - b'" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Difference a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Multiply:Expression{ |
|  | int value; |
|  | public: |
|  | Multiply(int a):value(a){} |
|  | void print(){ |
|  | cout << "a \* b"<< endl; |
|  | } |
|  | int calc(int a){ |
|  | return value \* a; |
|  | } |
|  | void derived(){ |
|  | cout << "Multiply derived is a' \* b + a \* b'" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Multiply a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Division:Expression{ |
|  | int value; |
|  | public: |
|  | Division(int a):value(a){} |
|  | void print(){ |
|  | cout << "a/b" << endl; |
|  | } |
|  | int calc(int a){ |
|  | return value/a; |
|  | } |
|  | void derived(){ |
|  | cout << "Division derived is (a' \* b - a \* b')/(b\*b)" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Division a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Sinus:Expression{ |
|  | int value; |
|  | public: |
|  | Sinus(int a):value(a){} |
|  | void print(){ |
|  | cout << "sin(x)"<< endl; |
|  | } |
|  | int calc(int a){ |
|  | return sin(a); |
|  | } |
|  | void derived(){ |
|  | cout << "Sinus derived is cos(a)" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Sinus a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Cosinus:Expression{ |
|  | int value; |
|  | public: |
|  | Cosinus(int a):value(a){} |
|  | void print(){ |
|  | cout << "cos(x)"<< endl; |
|  | } |
|  | int calc(int a){ |
|  | return cos(a); |
|  | } |
|  | void derived(){ |
|  | cout << "Cosinus derived is -sin(a)" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Cosinus a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Exp:Expression{ |
|  | int value; |
|  | public: |
|  | Exp(int a):value(a){} |
|  | void print(){ |
|  | cout << "e^x"<< endl; |
|  | } |
|  | int calc(int a){ |
|  | return exp(a); |
|  | } |
|  | void derived(){ |
|  | cout << "Exp derived is e^a" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Exp a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | class Ln:Expression{ |
|  | int value; |
|  | public: |
|  | Ln(int a):value(a){} |
|  | void print(){ |
|  | cout << "Ln(x)" << endl; |
|  | } |
|  | int calc(int a){ |
|  | return log(a); |
|  | } |
|  | void derived(){ |
|  | cout << "Ln derived is 1/a" << endl; |
|  | } |
|  | Expression\* copy(){ |
|  | Ln a = \*this; |
|  | return &a; |
|  | } |
|  | }; |
|  |  |
|  | int main(){ |
|  | double y, x; |
|  | cout << "Constant" << endl << "Enter value: "; |
|  | cin >> y; |
|  | Constant c(y); |
|  |  |
|  | cout << "Enter argument: "; |
|  | cin >> x; |
|  |  |
|  | c.print(); |
|  | cout << "f(" << x << ") = " << c.calc(x) << endl; |
|  | c.derived(); |
|  |  |
|  | cout << endl << "Variable" << endl << "Enter argument: "; |
|  | cin >> x; |
|  | Variable v(x); |
|  | v.print(); |
|  | cout << "f(" << x << ") = " << v.calc(x) << endl; |
|  | v.derived(); |
|  |  |
|  | cout << endl << "Sum" << endl << "Enter x and y: "; |
|  | cin >> x >> y; |
|  | Sum s(y); |
|  | s.print(); |
|  | cout << "f(" << x << ") = " << s.calc(x) << endl; |
|  | s.derived(); |
|  |  |
|  | cout << endl << "Difference" << endl << "Enter x and y: "; |
|  | cin >> x >> y; |
|  | Difference d(y); |
|  | d.print(); |
|  | cout << "f(" << x << ") = " << d.calc(x) << endl; |
|  | d.derived(); |
|  |  |
|  | return 0; |
|  | } |

**Вывод**

Все поставленные задачи были выполнены. В ходе выполнения работы был изучен механизм наследования классов в C++. Также для выполнения работы были использованы абстрактные классы и чисто виртуальные методы базовых абстрактных классов. Результаты работы программ представлены на скриншотах (рис 19, 20). Исходный код программ также доступен на GitHub по ссылкам:

https://github.com/adanil/OOP/tree/master/Practica

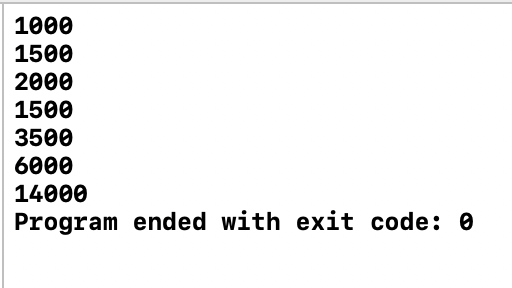


Рис. 7 Результат работы программы №1

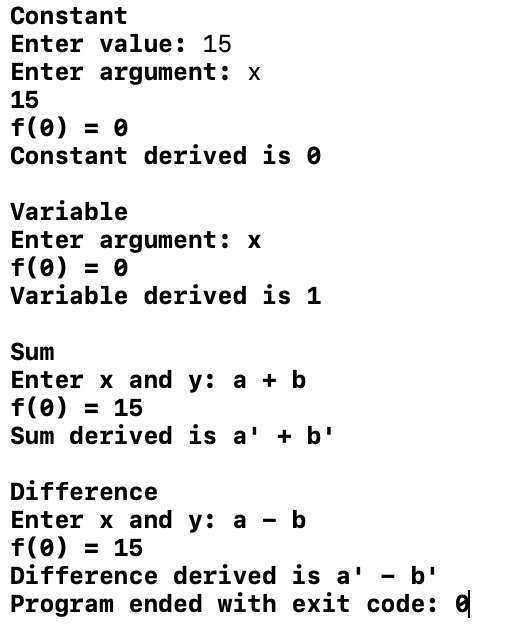


Рис. 8 Результат работы программы №2

## Практическая работа №7

**Абстрактные классы. Множественное наследование**

**Цель практической работы**

Целью данной практической работы является знакомство с абстрактными классами и механизмом виртуальных функций, а так же множественным наследованием на языке программирования C++.

**Задачи**

1. Реализовать абстрактный класс «Животное» и путём наследования от него получить классы «Кошка», «Собака», «Попугай».
2. Реализовать абстрактный класс «Фигура» и путём наследования от него получить абстрактный класс «Четырёхугольник», и затем путём наследования получить классы «Ромб», «Прямоугольник».
3. Реализовать абстрактный класс «Транспортное средство» и путём наследования от него получить классы «Автомобиль», «Автобус», «Велосипед».
4. Реализовать абстрактные классы «Экран» и «Клавиатура», путём наследования от них получить классы «Ноутбук», «Телефон», «Стационарный компьютер».

**Ход работы**

|  |
| --- |
| #include <iostream> |
|  | #include <string> |
|  | using namespace std; |
|  | //Вариант 1 |
|  | class Animal{ |
|  | protected: |
|  | string name; |
|  | int paws; |
|  | int years; |
|  | public: |
|  | virtual void abstract() = 0; |
|  | }; |
|  |  |
|  | class Cat:Animal{ |
|  | public: |
|  | virtual void abstract(); |
|  | }; |
|  |  |
|  | class Dog:Animal{ |
|  | public: |
|  | virtual void abstract(); |
|  | }; |
|  |  |
|  | class Parrot:Animal{ |
|  | public: |
|  | virtual void abstract(); |
|  | }; |
|  | //Вариант 2 |
|  |  |
|  | class Figure{ |
|  | protected: |
|  | double square; |
|  | int cntAngle; |
|  | public: |
|  | virtual double calcSquare() = 0; |
|  | }; |
|  |  |
|  | class Tetragon:Figure{ |
|  | public: |
|  | virtual double calcSquare(); |
|  | }; |
|  |  |
|  | class Rhombus:Tetragon{ |
|  | public: |
|  | virtual double calcSquare(); |
|  | }; |
|  | class Rectangle:Tetragon{ |
|  | public: |
|  | virtual double calcSquare(); |
|  | }; |
|  |  |
|  | //Вариант 3 |
|  | class Vehicle{ |
|  | protected: |
|  | string carNumber; |
|  | string model; |
|  | public: |
|  | virtual double calcFuel() = 0; |
|  | }; |
|  |  |
|  | class Car:Vehicle{ |
|  | public: |
|  | virtual double calcFuel() = 0; |
|  | }; |
|  |  |
|  | class Bus:Vehicle{ |
|  | public: |
|  | virtual double calcFuel() = 0; |
|  | }; |
|  |  |
|  | class Bicycle:Vehicle{ |
|  | public: |
|  | virtual double calcFuel() = 0; |
|  | }; |
|  |  |
|  | //Вариант 4 |
|  |  |
|  | class Computer{ |
|  | protected: |
|  | string model; |
|  | string processor; |
|  | public: |
|  | virtual void printModel() = 0; |
|  | }; |
|  |  |
|  | class Keyboard{ |
|  | protected: |
|  | string language; |
|  | string brand; |
|  | public: |
|  | virtual void printBrand() = 0; |
|  | }; |
|  |  |
|  | class Notebook:Computer,Keyboard{ |
|  | public: |
|  | virtual void printModel(); |
|  | virtual void printBrand(); |
|  | }; |
|  |  |
|  | class Phone:Computer,Keyboard{ |
|  | public: |
|  | virtual void printModel(); |
|  | virtual void printBrand(); |
|  | }; |
|  |  |
|  | class Desktop:Computer,Keyboard{ |
|  | public: |
|  | Desktop(string modl,string proc,string lang,string br){ |
|  | this->model = modl; |
|  | this->processor = proc; |
|  | this->language = lang; |
|  | this->brand = br; |
|  | } |
|  | virtual void printModel(){ |
|  | cout << model << endl; |
|  | } |
|  | virtual void printBrand(){ |
|  | cout << brand << endl; |
|  | } |
|  | }; |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | Desktop myComp("mac","intel","rus","apple"); |
|  | myComp.printBrand(); |
|  | myComp.printModel(); |
|  | return 0; |

**Вывод**

Все задачи были выполнены. В ходе выполнения работы были более плотно изучены множественное наследование и абстрактные классы. Для достижения поставленной цели были использованы различные STL контейнеры и библиотека cmath. Результаты работы программ представлены на скриншотах (рис. 21-24). Исходный кодвыполненных програм также доступе но ссылка на GitHub:

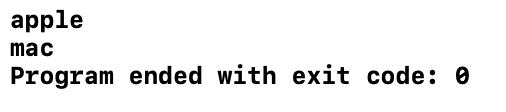
https://github.com/adanil/OOP/tree/master/Practica

Рис. 9 Результат работы программы

## Практическая работа №8

**Работа с файлами в языке С++**

**Цель практической работы**

Целью данной практической работы является приобретение практических навыков по работе с файлами на языке программирования языке C++.

**Задачи**

1. Реализуйте программу, считывающую текст из файла и выводящую каждое слово с новой строки.
2. Реализуйте программу, считывающую текст с клавиатуры и записывающую его в файл.

**Ход работы**

|  |
| --- |
| #include <iostream> |
|  | #include <fstream> |
|  | #include <string> |
|  | using namespace std; |
|  |  |
|  | int main(int argc, const char \* argv[]) { |
|  | //Вариант 1 |
|  | fstream file("/Users/daniilavtusko/Desktop/coding/Lab1OOP/Practica8/Practica8/test.txt"); |
|  | string s; |
|  | while (!file.eof()) { |
|  | file >> s; |
|  | cout << s << endl; |
|  | } |
|  |  |
|  | //Вариант 2 (Для прекращения ввода control + D или ctrl + z) |
|  | fstream fileOut("/Users/daniilavtusko/Desktop/coding/Lab1OOP/Practica8/Practica8/fileOut.txt"); |
|  | while (cin >> s){ |
|  | fileOut << s << endl; |
|  | } |
|  | return 0; |
|  | } |

**Вывод**

Все задачи были выполнены успешно. В ходе работы была исследована работа с файлами. Для достижения результата использовалась библиотек fstream, позволяющая производить работу с файлами. Результаты работ программ приведены на скриншотах (рис. 25, 26). Также исходный код програм доступенн по ссылкам на GitHub:

https://github.com/adanil/OOP/tree/master/Practica/Practica8

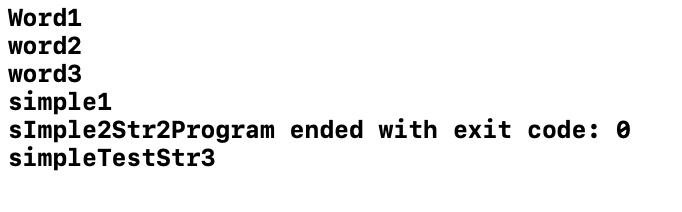


Рис. 10 Результат работы программы