**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**

Реализовать программу, которая считывает имя и здоровается.

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

**Ход работы**

Реализовать программу, которая считывает имя и здоровается

#include <iostream>

#include <string>

using namespace std;

int main() {

setlocale(LC\_ALL, "");

string a;

cout << "Name:";

cin >> a;

cout << "Hello, " << a << endl;

system("pause");

return 0;

}

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

#include <stdio.h>

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "");

float a,b;

cout << "Write a and b " << endl;

cin >> a >> b;

cout << a << "-" << b << "=" << a - b << endl;

cout << a << "+" << b << "=" << a + b << 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 рисунках (рис. 1 и 2). Также исходный код программ был помещён на GitHub и доступен по ссылкам:

1. https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%961

Рис. 1 Результат работы первой программы

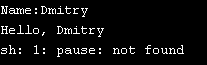
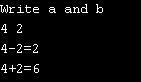


Рис. 2 Результат работы второй программы



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**

Вариант 1:

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

float a , b , n , m;

cout << "Write a and b" << endl;

cin >> a >> b;

if(a>=b){

n=pow(a-b,1/3.0);

}

else {

n=a\*a+(a-b)/sin(a\*b);

}

if(n<b){

m=((n+a)/-b)+pow(sin(a)\*sin(a)-cos(n),1/2.0);

}

else {

if(n==b){

m=b\*b + tan(n\*a);

}else{

m=b\*b\*b +n\*a\*a;

}

}

cout << " a=" << a << " b=" << b << " n=" << n << " m=" << m << endl;

return 0;

}

Вариант 2:

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

float a, b, z, t;

cout << "Write a and b" << endl;

cin >> a >> b;

if(a<b){

z=pow(abs(a\*a-b\*b),1/2.0);

}

else {

z=1-2\*cos(a)\*sin(a);

}

if(z<b){

t=pow(z+a\*a\*b,1/3.0);

}

else{

if(z==b){

t=1-log10(z)+cos(a\*a\*b);

}else{

t=1/cos(z\*a);

}

}

cout << " a=" << a << " b=" << b << " z=" << z << " t=" << t << endl;

return 0;

}

Вариант 3:

#include <iostream>

#include <cmath>

using namespace std;

int main() {

float a, b, y, t;

cout << "Enter a and b: ";

cin >> a >> b;

if (a <= b) {

y = (a-b)\*(a+b)/((a+b)\*(a\*a-a \*b+b\*b));

} else {

y = a + pow(log(b), 2);

}

if (y == b) {

t = (2\*y+sqrt(y\*y-a))/(2\*b-sqrt(a\*a-y));

} else {

if (y < b) {

t = sin(y)\*sin(y)+1/tan(a-b);

} else {

t = cbrt(y\*sin(a))+1/sqrt(y\*cos(b));

}

}

cout << " a = " << a << " b = " << b << " y = " << y << " t = " << t << endl;

system("pause");

return 0;

}

Вариант 4:

#include <iostream>

#include <cmath>

using namespace std;

int main() {

float a, x, y, t;

cout << "Enter a and x: ";

cin >> a >> x;

if (a<=x) {

y = a+log(x+a);

} else {

y = sqrt(sin(a\*x));

}

if (a>y) {

t = y/(a-x);

} else {

if (a==y) {

t =y/(a-x)+(a+x)/(y\*y);

} else {

t =tan(a\*x)+cos(2\*a\*y);

}

}

if (isnan(y)|| isnan(t))

cout << "error" << endl;

else

cout << " a = " << a << " x = " << x << " y = " << y << " t = " << t << 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 рисунках (рис. 3-6). Также код программ доступен по ссылкам:

1. https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%962

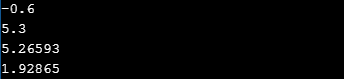


Рис. 3 Результат работы первой программы

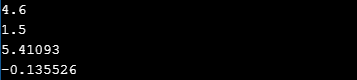


Рис. 4 Результат работы второй программы

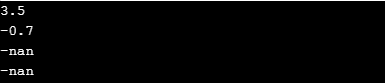


Рис. 5 Результат работы третьей программы

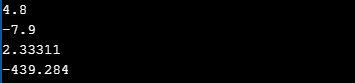


Рис. 6 Результат работы четвертой программы

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**

Реализовать класс «Книга»

Реализовать класс «Группа студентов»

Реализовать класс «Автомобиль»

Реализовать класс «Вектор»

Реализовать класс «Библиотека»

**Ход работы**

Класс «Книга»:

#include <iostream>

#include <string>

using namespace std;

class Book

{

string author;

string name;

int page;

public:

Book (string author, string name, int page)

{

setAuthor (author);

setName (name);

setPage (page);

}

void setAuthor (string author)

{

this->author=author;

}

void setName (string name)

{

this->name=name;

}

void setPage (int page)

{

this->page=page;

}

string getAuthor ()

{

return author;

}

string getName ()

{

return name;

}

int getPage ()

{

return page;

}

};

int main (){

string author,name;

int page;

cout << "Enter author " << endl;

cin >> author;

cout << "Enter name " << endl;

cin >> name;

cout << "Enter page " << endl;

cin >> page;

Book book (author,name,page);

cout << "author: " << book.getAuthor() << endl;

cout << "name: " << book.getName() << endl;

cout << "page: " << book.getPage() << endl;

return 0;

}

Класс «Группастудентов»:

#include <iostream>

#include <string>

using namespace std;

class Group

{

string code;

int course;

public:

Group(string code,int course)

{

setCode (code);

setCourse (course);

}

void setCode (string code)

{

this->code=code;

}

void setCourse ( int course)

{

this->course=course;

}

string getCode ()

{

return code;

}

int getCourse ()

{

return course;

}

};

int main (){

string code;

int course;

cout « "Enter code" « endl;

cin » code;

cout « "Enter course" « endl;

cin » course;

Group students(code,course);

cout « "code " « students.getCode() « endl;

cout « "course " « students.getCourse() « endl;

return 0;

}

Класс «Автомобиль»:

#include <iostream>

#include <string>

using namespace std;

class Car

{

string brand;

string color;

public:

Car(string brand,string color)

{

setBrand (brand);

setColor (color);

}

void setBrand (string brand)

{

this->brand=brand;

}

void setColor ( string color)

{

this->color=color;

}

string getBrand ()

{

return brand;

}

string getColor ()

{

return color;

}

};

int main (){

string brand;

string color;

cout « "Enter car brand" « endl;

cin » brand;

cout « "Enter car color" « endl;

cin » color;

Car name(brand,color);

cout « "brand " « name.getBrand() « endl;

cout « "color " « name.getColor() « endl;

return 0;

}

Класс «Вектор»:

#include <iostream>

#include <string>

using namespace std;

class Vector

{

double module;

double coordinate;

double x;

double y;

double z;

public:

Vector(double module,double coordinate)

{

setModule (module);

setCoordinate (coordinate);

}

void setModule (double module)

{

this->module=module;

}

void setCoordinate ( double coordinate)

{

this->coordinate=coordinate;

}

double getModule ()

{

return module;

}

double getCoordinate ()

{

return coordinate;

}

};

int main (){

double module;

double coordinate;

cout « "Enter vector module" « endl;

cin » module;

cout « "Enter vector coordinates" « endl;

cin » coordinate;

Vector name(module,coordinate);

cout « "module " « name.getModule() « endl;

cout « "coordinate " « name.getCoordinate() « endl;

return 0;

}

Класс «Библиотека»:

#include <iostream>

#include <string>

using namespace std;

class Library

{

int row;

int shelf;

public:

Library(int row,int shelf)

{

setRow (row);

setShelf (shelf);

}

void setRow (int row)

{

this->row=row;

}

void setShelf ( int shelf)

{

this->shelf=shelf;

}

int getRow ()

{

return row;

}

int getShelf ()

{

return shelf;

}

};

int main (){

int row;

int shelf;

cout << "Enter row in library" << endl;

cin >> row;

cout << "Enter a shelf in the library" << endl;

cin >> shelf;

Library name(row,shelf);

cout << "book on a row " << name.getRow() << endl;

cout << "book on the shelf " << name.getShelf() << 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 рисунках (рис. 7-11). Код программ также доступен на GitHub по ссылкам:

https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%963

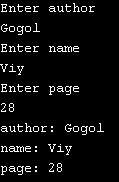


Рис. 7 Результат работы первой программы

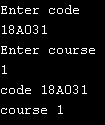


Рис. 8 Результат работы второй программы

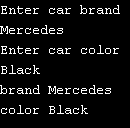


Рис. 9 Результат работы третьей программы

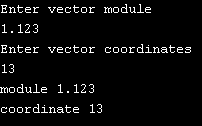


Рис 10 Результат работы четвертой программы

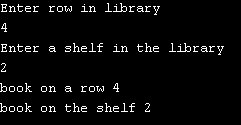


Рис. 11 Результат работы пятой программы

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**

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.

Объявите указатель на массив типа int и выделите память для 12-ти элементов. Необходимо написать функцию, которая поменяет значения четных и нечетных ячеек массива.

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**

Программа №1:

#include <iostream>

#include <ctime>

using namespace std;

double \*createArray(int);

void fillArray(double\*,int);

void showArray(double\*,int);

void clearArray(double\*,int);

int main() {

srand(time(NULL));

double \*Name;

int size = 0;

cout << "Enter array size (lower than 1 if you want to exit the program):" << endl;

cin >> size;

while(size>0){

Name = createArray(size);

fillArray(Name,size);

showArray(Name,size);

clearArray(Name,size);

cout << "Enter size of massive(lower than 1 if you want to exit the program): ";

cin >> size;

}

return 0;

}

double\* createArray(int size){

double \*GoodName = new double[size];

return GoodName;

}

void fillArray(double \*Array,int size){

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

Array[a]=rand() % 1000;

}

}

void showArray(double \*Array,int size){

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

cout << Array[a] << ' ';

}

}

void clearArray(double \*Array,int size){

delete[]Array;

}

Программа №2:

#include <iostream>

#include <ctime>

using namespace std;

void fillArr(int\* Arr, int sizeOfArr);

void showArr(int\* Arr, int sizeOfArr);

void changeElements(int\* Arr, int sizeOfArr);

int main (){

srand(time(NULL));

int sizeOfArr = 12;

int \*ptrArr=new int[sizeOfArr];

fillArr(ptrArr, sizeOfArr);

showArr(ptrArr, sizeOfArr);

changeElements(ptrArr, sizeOfArr);

showArr(ptrArr, sizeOfArr);

delete [] ptrArr;

return 0;

}

void fillArr(int\* ptrArr, int sizeOfArr)

{

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

ptrArr[i] = i \* 10;

}

}

void showArr(int\* ptrArr, int sizeOfArr)

{

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

{

cout << ptrArr[i] << " ";

}

cout << endl;

}

void changeElements(int\* ptrArr, int sizeOfArr)

{

for (int i = 0; i < sizeOfArr; i = i + 2) {

int a = ptrArr[i];

ptrArr[i] = ptrArr[i+1];

ptrArr[i+1] = a;

}

}

Программа №3:

#include <iostream>

#include <ctime>

using namespace std;

void showArray(int\*\* ptrArr, int stringAmount, int columnAmount);

void fillArray(int\*\* ptrArr, int strAmount, int colAmount);

int main()

{

srand(time(NULL));

int stringAmount = 0;

int columnAmount = 0;

cout << "Enter number of strings: ";

cin >> stringAmount;

cout << "Enter number of columns: ";

cin >> columnAmount;

int\*\* ptrArr = new int\*[stringAmount];

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

{

ptrArr[i]=new int[columnAmount];

}

fillArray(ptrArr, stringAmount, columnAmount);

showArray(ptrArr, stringAmount, columnAmount);

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

{

delete[] ptrArr[i];

}

delete[] ptrArr;

return 0;

}

void showArray(int\*\* ptrArr, int stringAmount, int columnAmount)

{

cout << endl;

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

{

cout << " ";

for (int j = 0; j < columnAmount; j++)

{

cout << ptrArr[i][j] << " ";

}

cout << " " << endl;

}

}

void fillArray(int\*\* ptrArr, int stringAmount, int columnAmount)

{

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

{

for (int j = 0; j < columnAmount; j++)

{

ptrArr[i][j] = rand() % 41 + 10;

}

}

}

**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 рисунках (рис.12-14). Так же исходный код программ доступен на GitHub по ссылкам:

1. https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%964

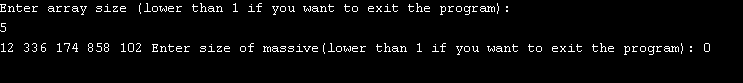


Рис. 12 Результат работы первой программы

pr4.2.PNG

Рис. 13 Результат работы второй программы

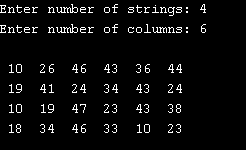


Рис. 14 Результат работы третьей программы

Практическая работа №5

**Перегрузка функций**

**Цель практической работы**

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

**Задачи**

Реализовать сортировку пузырьком для целых чисел, а затем перегрузить её для дробных.

Реализовать сортировку выбором для целых чисел, а затем перегрузить её для дробных.

Реализовать сортировку вставками для целых чисел, а затем перегрузить её для дробных.

Реализовать программу-калькулятор, работающую с разными типами данных.

**Ход работ**

Программа №1:

#include <iostream>

using namespace std;

void bubble\_sort(int \* array, int length) {

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

for (int j = i + 1; j < length;j++) {

if (array[i] > array[j]) {

swap(array[i],array[j]);

}

}

}

}

void bubble\_sort(float \* array, int length) {

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

for (int j = i + 1; j < length;j++) {

if (array[i] > array[j]) {

swap(array[i],array[j]);

}

}

}

}

int main() {

int array\_int[10] = {10,5,4,2,1,9,8,7,6,3};

float array\_f[10] = {10,5,4,2,1,9,8,7,6,3};

bubble\_sort(array\_int,10);

bubble\_sort(array\_f,10);

for (int i = 0; i < 10; i++) {

cout << array\_int[i] << '\t' << array\_f[i] << endl;

}

return 0;

}

Программа №2:

#include <iostream>

using namespace std;

void selection\_sort(int \* arr, int len) {

int start\_index = 0;

while (start\_index != len) {

int min\_index = start\_index;

for (int i = start\_index; i < len; i++) {

if (arr[min\_index] > arr[i]) {

min\_index = i;

}

}

swap(arr[start\_index],arr[min\_index]);

start\_index++;

}

}

void selection\_sort(float \* arr, int len) {

int start\_index = 0;

while (start\_index != len) {

int min\_index = start\_index;

for (int i = start\_index; i < len; i++) {

if (arr[min\_index] > arr[i]) {

min\_index = i;

}

}

swap(arr[start\_index],arr[min\_index]);

start\_index++;

}

}

int main() {

int array\_int[10] = {10,5,4,2,1,9,8,7,6,3};

float array\_f[10] = {10,5,4,2,1,9,8,7,6,3};

selection\_sort(array\_int,10);

selection\_sort(array\_f,10);

for (int i = 0; i < 10; i++) {

cout << array\_int[i] << '\t' << array\_f[i] << endl;

}

return 0;

}

Программа №3:

#include <iostream>

using namespace std;

void insertion\_sort(int \* arr, int len) {

int temp, item;

for (int counter = 1; counter < len; counter++)

{

temp = arr[counter];

item = counter-1;

while(item >= 0 && arr[item] > temp)

{

arr[item + 1] = arr[item];

arr[item] = temp;

item--;

}

}

}

void insertion\_sort(float \* arr, int len) {

int temp, item;

for (int counter = 1; counter < len; counter++)

{

temp = arr[counter];

item = counter-1;

while(item >= 0 && arr[item] > temp)

{

arr[item + 1] = arr[item];

arr[item] = temp;

item--;

}

}

}

int main() {

int array\_int[10] = {10,5,4,2,1,9,8,7,6,3};

float array\_f[10] = {10,5,4,2,1,9,8,7,6,3};

insertion\_sort(array\_int,10);

insertion\_sort(array\_f,10);

for (int i = 0; i < 10; i++) {

cout << array\_int[i] << '\t' << array\_f[i] << endl;

}

return 0;

}

Программа №4:

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

double calc(int a, int b);

double calc(double a, double b);

int main() {

int z;

cout << "Enter the type of numbers(1-Whole,2-fractional): " << endl;

cin >> z;

if (z == 1) {

int a , b ;

cout << "Enter two numbers (preferably whole): " << endl;

cin >> a >> b;

cout << "Answer: " << endl;

cout << calc(a, b) << endl;

}

else {

double a, b;

cout << "Enter two numbers (preferably fractional): " << endl;

cin >> a >> b;

cout << "Answer: " << endl;

cout << calc(a, b) << endl;

}

system("pause");

return 0;

}

double calc(int a, int b)

{

int k;

do {

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl;

cout << "Select operation..." << endl;

cout << "{} = I1+2-3\*4/" << endl;

cin >> k;

switch (k)

{

case 1:

return a + b;

break;

case 2:

return a - b;

break;

case 3:

return a\*b;

break;

case 4:

return a / b;

default:

break;

}

} while (k != 0);

}

double calc(double a, double b)

{

int k;

do {

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_" << endl;

cout << "Select operation..." << endl;

cout << "{} = D1+2-3\*d/" << endl;

cin >> k;

switch (k)

{

case 1:

return a + b;

break;

case 2:

return a - b;

break;

case 3:

return a\*b;

break;

case 4:

return a / b;

default:

break;

}

} while (k != 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 рисунках (рис. 15-18). Исходный код программ так же доступен на GitHub по ссылкам:

https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%965

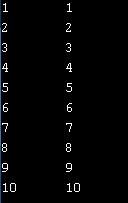


Рис. 15 Результат работы первой программы

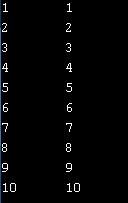


Рис. 16 Результат работы второй программы

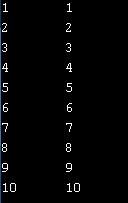


Рис. 17 Результат работы третьей программы

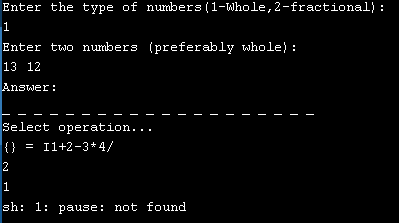


Рис. 18 Результат работы четвертой программы

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**

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.

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**

Программа №1:

#include <iostream>

using namespace std;

class Property {

protected:

int worth;

public:

Property(){}

virtual int result() = 0;

};

class Appartment: public Property {

public:

Appartment (int a) {worth = a;}

int result(){

return worth / 1000;

}

};

class Car: public Property {

public:

Car (int a) {worth = a;}

int result(){

return worth / 200;

}

};

class CountryHouse: public Property {

public:

CountryHouse(int a) {worth = a;}

int result (){

return worth / 500;

}

};

int main() {

Property \*\*array = new Property\*[7];

array[0] = new Appartment(1000000);

array[1] = new Appartment(200000);

array[2] = new Appartment(30000);

array[3] = new Car(10000);

array[4] = new Car(2000);

array[5] = new CountryHouse(1000000);

array[6] = new CountryHouse(200000);

for (int i = 0; i < 7; i++) {

cout << "The tax on this property is equal to: " << '\t' << array[i]->result() << endl;

}

for (int i = 0; i < 7; i++) {

delete array[i];

}

delete [] array;

return 0;

}

Программа №2:

#include <iostream>

#include <cmath>

using namespace std;

class Function {

public:

virtual void printFunction() = 0;

virtual double count(double x) = 0;

virtual void takeDerivative() = 0;

};

class Const : public Function{

double c;

public:

Const(double c) { this->c = c; }

void printFunction() { cout << "Function: f(x) = " << c << endl; }

double count(double x) { return c; }

void takeDerivative() { cout << "Derivative: 0" << endl; }

};

class Variable : public Function {

public:

void printFunction() { cout << "Function: f(x) = x" << endl; }

double count(double x) { return x; }

void takeDerivative() { cout << "Derivative: 1" << endl; }

};

class Sum : public Function {

double y;

public:

Sum(double y) { this->y = y; }

void printFunction() { cout << "Function: f(x) = x + y" << endl; }

double count(double x) { return x + y; }

void takeDerivative() { cout << "Derivative: 2" << endl; }

};

class Difference : public Function {

double y;

public:

Difference(double y) { this->y = y; }

void printFunction() { cout << "Function: f(x) = x - y" << endl; }

double count(double x) { return x - y; }

void takeDerivative() { cout << "Derivative: 0" << endl; }

};

class Product : public Function {

double y;

public:

Product(double y) { this->y = y; }

void printFunction() { cout << "Function: f(x) = x \* y" << endl; }

double count(double x) { return x \* y; }

void takeDerivative() { cout << "Derivative: y \* dx" << endl; }

};

class Quotient : public Function {

double y;

public:

Quotient(double y) { this->y = y; }

void printFunction() { cout << "Function: f(x) = x / y" << endl; }

double count(double x) { return x / y; }

void takeDerivative() { cout << "Derivative: dx / y" << endl; }

};

class Sin : public Function {

public:

void printFunction() { cout << "Function: f(x) = sin(x)" << endl; }

double count(double x) { return sin(x); }

void takeDerivative() { cout << "Derivative: cos(x)" << endl; }

};

class Cos : public Function {

public:

void printFunction() { cout << "Function: f(x) = cos(x)" << endl; }

double count(double x) { return cos(x); }

void takeDerivative() { cout << "Derivative: -sin(x)" << endl; }

};

class Exp : public Function {

public:

void printFunction() { cout << "Function: f(x) = e^x" << endl; }

double count(double x) { return exp(x); }

void takeDerivative() { cout << "Derivative: e^x" << endl; }

};

class Ln : public Function {

public:

void printFunction() { cout << "Function: f(x) = ln(x)" << endl; }

double count(double x) { return log(x); }

void takeDerivative() { cout << "Derivative: 1/x" << endl; }

};

int main() {

double y, x;

cout << "Constant" << endl << "Enter value: ";

cin >> y;

Const c(y);

cout << "Enter argument: ";

cin >> x;

c.printFunction();

cout << "f(" << x << ") = " << c.count(x) << endl;

c.takeDerivative();

cout << endl << "Variable" << endl << "Enter argument: ";

cin >> x;

Variable v;

v.printFunction();

cout << "f(" << x << ") = " << v.count(x) << endl;

v.takeDerivative();

cout << endl << "Sum" << endl << "Enter x and y: ";

cin >> x >> y;

Sum s(y);

s.printFunction();

cout << "f(" << x << ") = " << s.count(x) << endl;

s.takeDerivative();

cout << endl << "Difference" << endl << "Enter x and y: ";

cin >> x >> y;

Difference d(y);

d.printFunction();

cout << "f(" << x << ") = " << d.count(x) << endl;

d.takeDerivative();

cout << endl << "Product" << endl << "Enter x and y: ";

cin >> x >> y;

Product p(y);

p.printFunction();

cout << "f(" << x << ") = " << p.count(x) << endl;

p.takeDerivative();

cout << endl << "Quotient" << endl << "Enter x and y: ";

cin >> x >> y;

Quotient q(y);

q.printFunction();

cout << "f(" << x << ") = " << q.count(x) << endl;

q.takeDerivative();

cout << endl << "Sin" << endl << "Enter argument: ";

cin >> x;

Sin si;

si.printFunction();

cout << "f(" << x << ") = " << si.count(x) << endl;

si.takeDerivative();

cout << endl << "Cos" << endl << "Enter argument: ";

cin >> x;

Cos co;

co.printFunction();

cout << "f(" << x << ") = " << co.count(x) << endl;

co.takeDerivative();

cout << endl << "Exp" << endl << "Enter argument: ";

cin >> x;

Exp e;

e.printFunction();

cout << "f(" << x << ") = " << e.count(x) << endl;

e.takeDerivative();

cout << endl << "Ln" << endl << "Enter argument: ";

cin >> x;

Ln l;

l.printFunction();

cout << "f(" << x << ") = " << l.count(x) << endl;

l.takeDerivative();

system("pause");

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 рисунках (рис 19, 20). Исходный код программ также доступен на GitHub по ссылкам:

https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%966

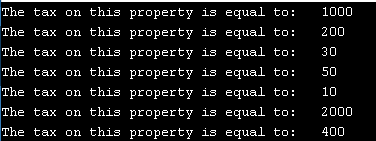


Рис. 19 Результат работы первой программы

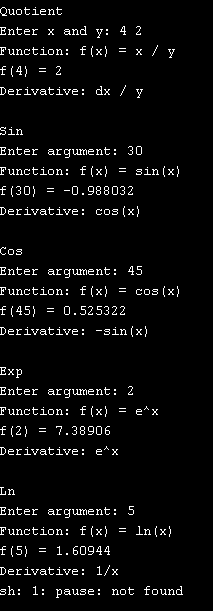
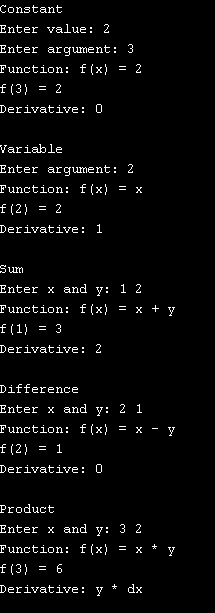


Рис. 20 Результат работы второй программы

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**

Реализовать абстрактный класс «Животное» и путём наследования от него получить классы «Кошка», «Собака», «Попугай».

Реализовать абстрактный класс «Фигура» и путём наследования от него получить абстрактный класс «Четырёхугольник», и затем путём наследования получить классы «Ромб», «Прямоугольник».

Реализовать абстрактный класс «Транспортное средство» и путём наследования от него получить классы «Автомобиль», «Автобус», «Велосипед».

Реализовать абстрактные классы «Экран» и «Клавиатура», путём наследования от них получить классы «Ноутбук», «Телефон», «Стационарный компьютер».

**Ход работы**

Программа №1:

#include <iostream>

#include <string>

using namespace std;

class Animal {

protected:

int size;

string name;

int weight;

virtual void make\_sound() = 0;

public:

Animal(int size, int weight,string name) {

this->name = name;

this->size = size;

this->weight = weight;

}

};

class Cat : public Animal {

public:

Cat( int size, int weight, string name) : Animal(size, weight, name) {

};

void make\_sound() {

cout << "Meow" << endl;

}

};

class Dog : public Animal {

public:

Dog( int size, int weight, string name) : Animal(size, weight, name) {

};

void make\_sound() {

cout << "Woof" << endl;

}

};

class Bird : public Animal {

public:

Bird( int size, int weight, string name) :Animal(size, weight, name) {

};

void make\_sound() {

cout << "Tweet" << endl;

}

};

int main (){

Cat a(2,2,"cat");

a.make\_sound();

Dog b(4,4,"dog");

b.make\_sound();

Bird c(1,1,"bird");

c.make\_sound();

return 0;

}

Программа №2:

#include <iostream>

#include <cmath>

using namespace std;

class Shape{

protected:

int a1, a2, a3, a4;

public:

Shape() {

cout << "Enter the length of the sides: " << endl;

cin >> a1;

cin >> a2;

cin >> a3;

cin >> a4;

}

virtual void square() = 0;

};

class Romb:public Shape {

public:

double sinus;

Romb() :Shape() {

cout << "Enter a smaller angle: " << endl;

cin >> sinus;

}

void square() {

cout << a1\*a1\*sin(sinus)<<endl;;

}

};

class Rectangle :public Shape {

public:

Rectangle() :Shape() {

}

void square() {

cout<< a1\*a3<<endl;

}

};

class Figure {

public:

void square(Shape \*shape) {

cout << "Area is equal to: ";

shape->square();

}

};

int main()

{

Rectangle c;

Romb b;

Figure a;

a.square(&b);

a.square(&c);

system("pause");

return 0;

}

Программа №3:

#include <iostream>

#include <cmath>

#include <string>

using namespace std;

class Transport {

protected:int weight;

int speed;

string name;

public:Transport() {

cout << "enter the mass and speed of the transport:" << endl;

cin >> weight >> speed;

cout << "enter the name of the transport" << endl;

cin >> name;

}

virtual void showAdvantages() = 0;

};

class Bycicle:public Transport {

public:

Bycicle() :Transport() { }

void showAdvantages() {

cout << "Bycicle:" << name << ' ' << speed << "km/h " << weight << "kg"<<endl;

}

};

class Bus :public Transport {

public:

Bus() :Transport() {}

void showAdvantages() {

cout << "Bus:" << name << ' ' << speed << "km/h " << weight << "kg" << endl;

}

};

class Car :public Transport {

public:

Car() :Transport() { }

void showAdvantages() {

cout << "Car:" << name << ' ' << speed << "km/h " << weight << "kg" << endl;

}

};

int main()

{

string b;

cout << "Enter the transport (Bicycle, Bus, Car): " << endl;

cin >> b;

if(b=="Bus"){ Bus a; a.showAdvantages();

}

else if (b == "Car") {

Car a; a.showAdvantages();

}

else if (b == "Bicycle") {

Bycicle a; a.showAdvantages();

}

system("pause");

return 0;

}

Программа №4:

#include <iostream>

using namespace std;

class Screen {

protected:

double screenSize;

public:

Screen(double size) { this->screenSize = size; }

virtual void changeScreen(double size) = 0;

};

class Keyboard {

protected:

int buttons;

public:

Keyboard(int buttons) { this->buttons = buttons; }

virtual void changeKeyboard(int buttons) = 0;

};

class Computer : public Screen, Keyboard {

public:

Computer(double screenSize, int buttons) : Screen(screenSize), Keyboard(buttons) {};

void changeScreen(double size) {

cout << "screen " << screenSize <<

" has been changed " << size << " size" << endl;

screenSize = size;

}

void changeKeyboard(int buttons) {

cout << "keyboard with " << this->buttons << " buttons" <<

" has been changed " << buttons << " buttons" << endl;

this->buttons = buttons;

}

~Computer() {}

};

class Notebook : public Screen, Keyboard {

public:

Notebook(double screenSize, int buttons) : Screen(screenSize), Keyboard(buttons) {};

void changeScreen(double size) {

if (screenSize == size) {

cout << "Screen has been changed." << endl;

}

else {

cout << "Screen not changed." << endl;

}

}

void changeKeyboard(int buttons) {

if (this->buttons == buttons) {

cout << "The keyboard has been changed." << endl;

}

else {

cout << "The keyboard has not been changed." << endl;

}

}

~Notebook() {}

};

class Phone : public Screen, Keyboard {

public:

Phone(double screenSize, int buttons) : Screen(screenSize), Keyboard(buttons) {};

void changeScreen(double size) {

if (screenSize == size) {

cout << "phone screen changed" << endl;

}

else {

cout << "phone screen has not been changed" << endl;

}

}

void changeKeyboard(int buttons) {

cout << "Keyboard cannot be changed!" << endl;

}

~Phone() {}

};

int main() {

int z=1;

do {

cout << "What device will work: 1-phone, 2- notebook, 3- computer" << endl;

int x, y;

cin >> z;

switch (z)

{

case 1: {cout << "Enter the screen size and number of buttons: " << endl;

cin >> x >> y;

Phone a(x, y);

cout << "How much do you want to change the screen size and the number of buttons: " << endl;

cin >> x >> y;

a.changeKeyboard(x);

a.changeScreen(y);

break; }

case 2: {cout << "Enter the screen size and number of buttons: " << endl;

cin >> x >> y;

Notebook b(x, y);

cout << "How much do you want to change the screen size and the number of buttons: " << endl;

cin >> x >> y;

b.changeKeyboard(x);

b.changeScreen(y);

break; }

case 3: {cout << "Enter the screen size and number of buttons: " << endl;

cin >> x >> y;

Computer c(x, y);

cout << "How much do you want to change the screen size and the number of buttons: " << endl;

cin >> x >> y;

c.changeKeyboard(x);

c.changeScreen(y);

break; }

default:

break;

}

} while (z != 0);

system("pause");

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 изображениях (рис. 21-24). Исходный код выполненных программ также доступе но ссылка на GitHub:https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%967

pr7.1.PNG

Рис. 21 Результат работы первой программы

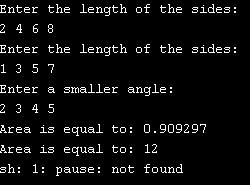


Рис. 22 Результат работы второй программы

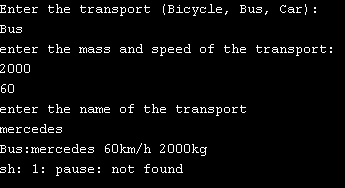


Рис. 23 Результат работы третьей программы

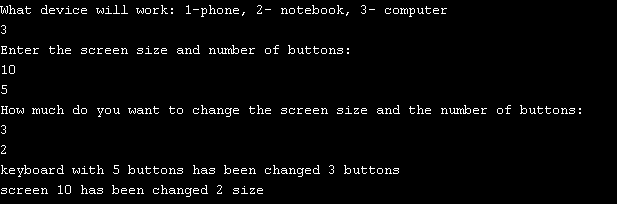


Рис. 24 Результат работы четвертой программы

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**

Реализуйте программу, считывающую текст из файла и выводящую каждое слово с новой строки.

Реализуйте программу, считывающую текст с клавиатуры и записывающую его в файл.

**Ход работы**

Программа №1:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ofstream creating\_file;

ifstream reading\_file;

reading\_file.open("file.txt");

if (reading\_file.is\_open()){

string a;

while(!reading\_file.eof()){

reading\_file >> a;

cout << a << ' ';

}

reading\_file.close();

}else{

cout << "File not found.Please write the data to a file.";

creating\_file.open("file.txt",ios\_base::trunc);

creating\_file.close();

}

return 0;

}

Программа №2:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ofstream file;

string a;

cout << "Enter text from the keyboard: "<< endl;

getline(cin,a);

file.open("file.txt", ios\_base::trunc);

file << a;

file.close();

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 рисунках (рис. 25, 26). Также исходный код програм доступен по ссылкам на GitHub:

1. https://github.com/ivankachura/OOP/tree/master/%D0%9F%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%E2%84%968

pr8.1.1.PNG pr8.2.2.PNG

Рис. 25 Результат работы первой программы

pr8.2.1.PNG pr8.2.2.PNG

Рис. 26 Результат работы второй программы