МИНИСТЕРСТВО ТРАНСПОРТА И КОММУНИКАЦИЙ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Белорусский государственный университет транспорта»

Кафедра «Информационно-управляющие системы и технологии»

Отчет по лабораторным работам

по дисциплине «Объектно-ориентированное программирование»

Выполнил Проверил

студент группы ГИ-21 ст. преп. каф. «ИУСиТ»

Зайцев И. С. Голдобина Т. А.

Гомель, 2020

Оглавление

[Лабораторная работа № 1 Тема: «Организация ввода/вывода, динамическое выделение памяти» 3](#_Toc59141053)

[Задания 3](#_Toc59141054)

[Контрольные вопросы 6](#_Toc59141055)

[Лабораторная работа № 2 Тема: «Классы и объекты» 8](#_Toc59141056)

[Задания 8](#_Toc59141057)

[Контрольные вопросы 9](#_Toc59141058)

[Лабораторная работа № 3 Тема: «Дружественные функции и классы» 12](#_Toc59141059)

[Задания 12](#_Toc59141060)

[Контрольные вопросы 14](#_Toc59141061)

[Лабораторная работа № 4 Тема: «наследование, простое наследование» 16](#_Toc59141062)

[Задания 16](#_Toc59141063)

[Контрольные вопросы 19](#_Toc59141064)

[Лабораторная работа № 5 Тема: «принцип полиморфизма, виртуальные функции, абстрактные классы» 20](#_Toc59141065)

[Задания 20](#_Toc59141066)

[Контрольные вопросы 23](#_Toc59141067)

[Лабораторная работа № 6 Тема: «множественное наследование, виртуальное наследование» 25](#_Toc59141068)

[Задания 25](#_Toc59141069)

[Контрольные вопросы 29](#_Toc59141070)

[Лабораторная работа № 7 Тема: «Шаблоны классов» 31](#_Toc59141071)

[Задания 31](#_Toc59141072)

[Контрольные вопросы 35](#_Toc59141073)

[Лабораторная работа № 8 Тема: «Обработка исключений» 36](#_Toc59141074)

[Задания 36](#_Toc59141075)

[Контрольные вопросы 40](#_Toc59141076)

[Лабораторная работа № 9 Тема: «Разработка приложений с использованием типа String.» 42](#_Toc59141077)

[Задания 42](#_Toc59141078)

[Контрольные вопросы 44](#_Toc59141079)

# Лабораторная работа № 1 Тема: «Организация ввода/вывода, динамическое выделение памяти»

### **Цель**

Изучить организацию ввода/вывода и работу с динамической памятью при программировании алгоритмов в C++.

## Задания

### Задание 1.1

### Условие

Реализовать функции для работы с матрицами:

а) функция производит перемножение матриц;

б) функция производит сложение двух матриц.

Память под матрицы выделять динамически. Необходимо освобождать память, выделенную под матрицы.

Программный код

#include <iostream>

#include <cstdlib>

using namespace std;

int addition()

{

int row1, row2, col1, col2, i, j;

cout << "\nВведите количество строк первой матрицы(сложение): ";

cin >> row1;

cout << "Введите количество столбцов первой матрицы(сложение): ";

cin >> col1;

cout << "Введите количество строк второй матрицы(сложение): ";

cin >> row2;

cout << "Введите количество столбцов второй матрицы(сложение): ";

cin >> col2;

int Mass[row1][col1], Mass2[row2][col2], MassRes[row1][col2];

if (row1 != row2 || col1 != col2)

{

cout << "\nСложение невозможно\n";

return 0;

}

for (i=0; i<row1; i++)

{

for (j=0; j<col1; j++)

{

Mass[i][j]=rand() % 25;

cout << Mass[i][j] << "\t";

}

cout << endl;

}

cout << "\n";

for (i=0; i<row2; i++)

{

for (j=0; j<col2; j++)

{

Mass2[i][j]=rand() % 25;

cout << Mass2[i][j] << "\t";

}

cout << endl;

}

cout << "\nРезультат сложения:\n";

for (i=0; i<row1; i++)

{

for(j=0; j<col2; j++)

{

MassRes[i][j]=Mass[i][j]+Mass2[i][j];

cout << MassRes[i][j] << "\t";

}

cout << endl;

}

}

int multiplication()

{

int row1, row2, col1, col2, i, j;

cout << "\nВведите количество строк первой матрицы(умножение): ";

cin >> row1;

cout << "Введите количество столбцов первой матрицы(умножение): ";

cin >> col1;

cout << "Введите количество строк второй матрицы(умножение): ";

cin >> row2;

cout << "Введите количество столбцов второй матрицы(умножение): ";

cin >> col2;

if (col1 != row2)

{

cout << "Умножение невозможно!";

return 0;

}

int Mass[row1][col1], Mass2[row2][col2], MassRes[row1][col2];

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

Mass[i][j]=rand() %50;

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

}

cout << endl;

}

cout << "\n";

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

Mass2[i][j]=rand() %50;

cout << Mass2[i][j] << " ";

}

cout << endl;

}

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col2; j++)

{

MassRes[i][j] = 0;

for (int k = 0; k < col1; k++)

MassRes[i][j] += Mass[i][k] \* Mass2[k][j];

}

}

cout << "Матрица произведения:" << endl;

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col2; j++)

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

cout << endl;

}

}

int main()

{ int key;

setlocale(LC\_ALL,"");

cout << "Сложение матриц(1);\nУмножение матриц(2);\nВыход(0);";

while(key!=0)

{

cout << "\nВведите номер: ";

cin >> key;

if(key==1)

{

addition();

}

else if(key==2)

{

multiplication();

}

else{return 0;}

}

return 0;

}

### Результат

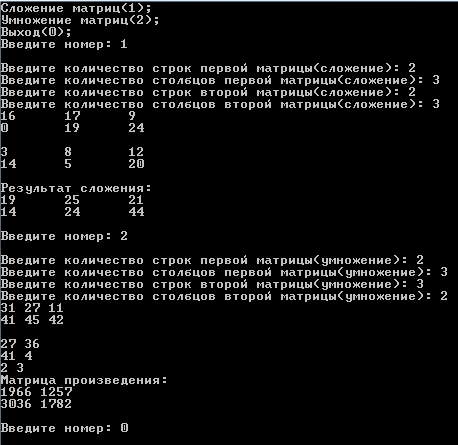


Рисунок 1 – Результат задания 1

## Контрольные вопросы

1. Как осуществляется ввод/вывод в C++?

В C++ ввод и вывод данных производится потоками байт. Поток (последовательность байт) – это логическое устройство, которое выдает и принимает информацию от пользователя и связано с физическими устройствами ввода/вывода. При операциях ввода байты направляются от устройства в основную память. В операциях вывода – наоборот.

Для ввода информации с клавиатуры используется поток cin, для вывода – cout.

1. Что такое манипулятор ввода/вывода.

Манипулятор – объект особого типа, который управляет потоками ввода/вывода, для форматирования передаваемой в потоки информации. Манипуляторы помещаются в операторы ввода/вывода непосредственно перед форматируемым значением.

1. Для чего необходимы операторы new и delete? В чем их отличие от функций malloc() и free()?

Операторы new и delete используют для выделения и освобождения памяти соответственно.

С указателями на объект нельзя использовать функции выделения и освобождения динамической памяти malloc() и free() , так как выделение памяти под объект не вызывает его конструктор, в результате возникает неинициализированный объект, который ведет себя непредсказуемо, поэтому в C++ используются операторы new и delete.

1. Как создать и удалить массив объектов?

Для создания массива объектов используется оператор new, объявляется следующим образом:

new имя\_типа [выражение];

Для удаления используется оператор delete, объявляется следующим образом:

delete [ ] переменная\_указатель;

1. Как можно выделить память под двумерный массив?

Используя «указатель на указатель», в языке C++ допустимо объявлять переменные, имеющие тип «указатель на указатель». Объявляется он следующим образом:

int \*\*mass;

### Вывод по практической работе

Изучил организацию ввода/вывода и работал с динамической памятью при программировании алгоритмов в C++.

# Лабораторная работа № 2 Тема: «Классы и объекты»

### **Цель**

Изучить структуру класса, атрибуты доступа к компонентам класса; рассмотреть принцип работы конструкторов (с параметрами, с параметрами по умолчанию), конструктора копирования и деструктора при работе с объектом, статические и константные данные и методы.

## Задания

### Задание 1.2

### Условие

Создать класс «Окружность». Данные класса: радиус окружности.

Функции-члены класса: вычисление площади, вычисление длины окружности,

установление радиуса окружности и возвращение значения радиуса

окружности, вывод данных на экран. Функции-члены установки полей класса

должны проверять корректность задаваемых параметров.

Программный код

#include <iostream>

#include <math.h>

#define PI 3.141593

using namespace std;

class Circle{

public:

float r;

Circle(){

cout << "Enter the radius of the circle: ";

cin >> r;

};

int check()

{

if(r<=0)

{

cout << "Error!!!\n";

return -1;

}

}

float area()

{

float s;

s=PI\*pow(r,2);

cout << "Area = " << s <<endl;

return s;

}

float length()

{

float p;

p=2\*PI\*r;

cout << "Length = " << p <<endl;

return p;

}

};

int main()

{ int a;

Circle circle = Circle();

a=circle.check();

if(a==-1)

{

return -1;

}

else

{

circle.area();

circle.length();

}

return 0;

}

### Результат



Рисунок 2 – Результат задания 1

## Контрольные вопросы

1. В чем разница между struct, class и union?

В языке С++ класс, определяемый посредством ключевых слов struct, class, union, включает в себя функции и данные, создавая новый тип объектов. Компоненты класса имеют ограничения на доступ. Эти ограничения определяются ключевыми словами private, protected, public. Для ключевого слова class по умолчанию все компоненты будут private. Элементы класса типа структуры (struct) и объединения (union) по умолчанию принимаются как public.

1. Что такое указатель this? Приведите пример использования этого указателя.

Указатель this – скрытый указатель, который адресует начало выделенной под объект памяти. Для любой функции, принадлежащей классу my\_class, указатель this неявно объявлен так:

my\_class \*const this;

Пример:

class Something {

private:

    int data;

public:

    Something(int data) {

        this->data = data;  }};

1. Какова основная форма конструктора копирования и когда он вызывается?

Общий вид конструктора копирования имеет следующий вид:

имя\_класса (const имя\_класса & );

Когда объект копируется из функции main() в функцию, то должен вызываться конструктор для копирования. Т.е., если в конструкторе некоторого класса осуществляется динамическое выделение памяти, такой класс должен иметь соответствующий конструктор для копирования, а также деструктор (для освобождения памяти).

1. Когда вызывается деструктор?

Деструктор вызывается при уничтожении объекта.

1. Приведите пример использования константных и статических данных и методов.

#include <iostream>

using namespace std;

class cls

{

int kl; // количество изделий

double zp; // зарплата на производство одного изделия

double nl1,nl2; // два налога на зарплату

double sr; // количество сырья на производство одного изделия

static double cs; // цена сырья на одно изделие

public:

cls(){} // конструктор по умолчанию

~cls(){} // деструктор

void inpt(int);

static void vvod\_cn(double);

double seb() const;

};

double cls::cs; // явное определение static-члена в контексте файла

void cls::inpt(int k)

{

kl=k;

cin >> nl1 >> nl2 >> zp;

}

void cls::vvod\_cn(double c)

{

cs=c; // можно обращаться в функции только к static-компонентам

}

double cls::seb() const

{

return kl\*(zp+zp\*nl1+zp\*nl2+sr\*cs);

// в функции нельзя изменить ни один

// неявный параметр (kl zp nl1 nl2 sr)

}

int main()

{

setlocale(LC\_ALL,"Russian");

cls c1,c2;

cout << "Введите зарплату и два налога для объекта с1" << endl;

c1.inpt(100); // инициализация первого объекта

cout << "Введите зарплату и два налога для объекта с2" << endl;

c2.inpt(200); // инициализация второго объекта

cls::vvod\_cn(500.);

cout << "\nc1 = " << c1.seb() << "\nc2 = " << c2.seb() << endl;

return 0;

}

### Вывод по практической работе

Изучил структуру класса, атрибуты доступа к компонентам класса; рассмотрел принцип работы конструкторов (с параметрами, с параметрами по умолчанию), конструктора копирования и деструктора при работе с объектом, статические и константные данные и методы.

Лабораторная работа № 3  
Тема: «Дружественные функции и классы»

### **Цель**

Понять назначение дружественных функций и классов, изучить принципы перегрузки бинарных и унарных операций.

## Задания

### Задание 1.3

### Условие

Реализовать класс String для работы со строками символов. Перегрузить операторы «=», «+=» так, чтобы производилось сложение строки и объекта. Предоставить конструктор копирования. Определить friend-функции для операторов ввода/вывода в поток.

Программный код

#include <iostream>

#include <clocale>

using namespace std;

class String {

int size;

char\* mas;

public:

String(string str) {

this->size = str.size();

mas = new char[size];

for (int i = 0; i < size; i++) { mas[i] = str[i]; }

}

String(char\* mas, int size) {

this->size = size>0 ? size : 0;

this->mas = mas;

}

String operator = (string s) {

int newStrSize = this->size + s.size();

char newStr[newStrSize];

int i;

for (i = 0; i < this->size; i++) { newStr[i] = this->mas[i];}

for (int j = 0; i < newStrSize; i++, j++) {newStr[i] = s[j];}

return String(newStr, newStrSize);

}

String operator += (string s) {

int newStrSize = this->size + s.size();

char newStr[newStrSize];

int i;

for (i = 0; i < this->size; i++) { newStr[i] = this->mas[i];}

for (int j = 0; i < newStrSize; i++, j++) {newStr[i] = s[j];}

return String(newStr, newStrSize);

}

void print() {

for(int i = 0; i < size; i++) { cout << mas[i]; }

cout << endl;

}

friend std::ostream& operator<< (std::ostream &out, const String &string) {

for(int i = 0; i < string.size; i++) { out << string.mas[i]; }

out << endl;

}

friend std::istream& operator>> (std::istream &in, String &strin) {

in >> strin.size;

for (int i = 0; i < strin.size; i++) { in >> strin.mas[i];}

return in;

}

};

int main() {

setlocale(LC\_ALL, "");

char a[3] = {'a', 'b', 'c'};

String first(a, 3);

cout << first;

String second("hjhjbjb");

cin >> second;

cout << second;

(first = "dab").print();

cout << (second += "cad");

return 0;

}

### Результат

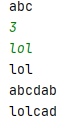


Рисунок 3 – Результат задания 1

## Контрольные вопросы

1. Почему может потребоваться перегрузка оператора присваивания?

Перегрузка оператора присваивания занимает важное место при реализации классов, использующих динамическую память. Она необходима, если нам присвоить полям экземпляра класса, поля другого экземпляра, либо инициировать их после объявления. Присваивание, так же, как и конструктор копии, по умолчанию реализуется побайтным копированием, которое подразумевает только копирование значений полей объекта. Поэтому при использовании динамической памяти в классе произойдет копирование ссылки на нее, а сама область динамической памяти станет разделяемой между двумя объектами. Такое разделение доступа к памяти в C++ при работе с указателями не является корректным.

1. Можно ли изменить приоритет перегруженного оператора?

Перегрузка операций подчиняется правилу, что сохраняются приоритеты операций и правила ассоциации (справа налево или слева направо) по сравнению с использованием в стандартных типах данных.

1. Когда следует переопределять операторы с помощью дружественных функций, а когда с помощью функций элементов класса?

Их нужно предопределять, когда мы хотим, чтобы при использовании с объектами конкретного класса они выполняли заданные функции, что дает возможность использовать собственные типы данных точно так же, как стандартные.

1. Назовите особенности дружественных функций.

Особенностей у дружеских функций достаточно много, но главные из них: дружеские функции не являются компонентами-функциями класса; имеют доступ ко всем компонентами этого класса (private, public, protected); дружеские функции не могут использовать указатель this; в общем случае, дружеские функции являются глобальными независимо от сектора, в котором объявлена (если не объявлена ни в одном другом классе без спецификатора friend); дружественные функции могут рассматриваться как часть интерфейса класса с внешней среды; дружественные функции не наследуются; отношение friend не является ни симметрическим, ни транзитивным, также друзьями класса можно определить перегружение функции.

1. Опишите особенности перегрузки постфиксных и префиксных операторов «++» и «--»

Особенности перегрузки этих операторов состоит в том, что нужно перегружать как префиксную, так и постфиксную форму этих операторов, дружественная к классу функция не получает указателя this класса, параметр в эту функцию может передаваться по ссылке, а не по значению, т. е. перед именем параметра указывается символ ссылки &. Вообще у префиксной формы определенных особенностей нет, в то время как постфиксная должна возвращать значение, которое было до его изменения.

### Вывод

В результате лабораторной работы научились перегружать операторы, изучили дружественные функции и классы.

Лабораторная работа № 4  
Тема: «наследование, простое наследование»

### **Цель**

Изучить принципы наследования.

## Задания

### Задание 1.4

### Условие

Написать программу, в которой описана иерархия классов: базовый класс – «ошибка в программе», производные классы - «ошибка доступа к памяти», «математическая», «деление на ноль», «переполнение». Наследники должны иметь поля, содержащие дополнительные сведения об ошибке, если такие имеются. Продемонстрировать работу всех методов классов, предоставив пользователю выбор типа объекта для демонстрации.

Программный код

#include <iostream>

using namespace std;

class Error\_programm

{

public:

string error;

Error\_programm()

{

error="ERROR!!!";

}

virtual void vivod()

{

cout <<error<<endl;

}

};

class Error\_memory : public Error\_programm

{ public:

string error\_inf="An attempt was made to free a heap that was not previously allocated.";

virtual void vivod()

{

Error\_programm::vivod();

cout <<error\_inf<<endl;

}

};

class Error\_math : public Error\_programm

{ public:

string error\_inf= "Sine and cosine range: -1 to 1";

virtual void vivod()

{

Error\_programm::vivod();

cout <<error\_inf<<endl;

}

};

class Error\_division : public Error\_programm

{ public:

string error\_inf= "ERROR: You cannot divide by zero :(";

virtual void vivod()

{

Error\_programm::vivod();

cout <<error\_inf<<endl;

}

};

class Error\_overflow : public Error\_programm

{ public:

string error\_inf= "ERROR: Buffer overflow";

virtual void vivod()

{

Error\_programm::vivod();

cout <<error\_inf<<endl;

}

};

int main()

{

setlocale(LC\_ALL,"");

int key;

cout <<"1.Error\_programm\n2.Error\_memory\n3.Error\_math\n4.Error\_division\n5.Error\_overflow\n0.Exit\n"<<endl;

do

{

cout << "Enter problem number: ";

cin >> key;

switch(key)

{ case 0: cout <<"Have you problem with your head?"<<endl; break;

case 1:

{Error\_programm error1; error1.vivod();

break;}

case 2:

{Error\_memory error2; error2.vivod();

break;}

case 3:

{Error\_math error3; error3.vivod();

break;}

case 4:

{Error\_division error4; error4.vivod();

break;}

case 5:

{Error\_overflow error5; error5.vivod();

break;}

}

}while(key!=0);

getchar();

return 0;

}

### Результат

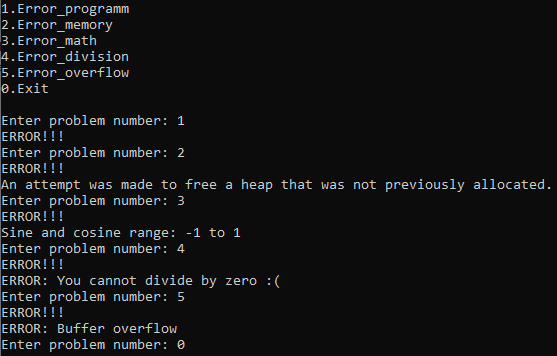


Рисунок 4 – Результат задания 1

## Контрольные вопросы

1. Опишите модификаторы доступа и наследования. Как изменяются атрибуты доступа элементов класса при наследовании?

Атрибут класса (модификатор прав доступа) может задаваться ключевыми словами public и private. Атрибут может опускаться – в этом случае принимается атрибут по умолчанию (для ключевого слова class – private, для struct – public).

|  |  |  |
| --- | --- | --- |
| Ограничения на доступ в базовом классе | Модификатор наследования прав | Ограничения на доступ в производном классе |
| Private | private | нет доступа |
| Protected | private | private |
| Public | private | private |
| Private | public | нет доступа |
| Protected | public | protected |
| Public | public | public |

В производных классах права на доступ к элементам базовых классов могут быть только ограничены, а не расширены.

1. Как работают конструкторы при наследовании?

Конструкторы не наследуются, поэтому производный класс должен иметь собственные конструкторы. Если у базового и произвольного классов имеются конструкторы, то они выполняются в порядке наследования. Если конструктор базового класса требует указания параметров, он должен быть явным образом вызван в конструкторе производного класса в списке инициализации.

1. Как работают деструкторы при наследовании?

Деструкторы не наследуются. Если деструктор не был описан в производном классе, то он формируется по умолчанию и вызывает деструкторы всех базовых классов. В деструкторе производного класса не требуется явно вызывать деструкторы базовых классов, т.к. это будет сделано автоматически. Порядок вызова деструкторов – обратный вызову конструкторов.

1. Какой класс называется производным?

Производным классом называют класс происходящий от базового.

1. Как объявляются производные классы?

ключ\_класса имя\_производного\_класса :

необязательный\_модификатор\_доступа имя\_базового\_класса {

Тело производного класса };

### Вывод

В результате лабораторной работы изучили принципы иерархии классов.

Лабораторная работа № 5  
Тема: «принцип полиморфизма, виртуальные функции, абстрактные классы»

### **Цель**

Изучить реализацию принципа полиморфизма через использование виртуальных функций при наследовании.

## Задания

### Задание 1.5

### Условие

Создать класс Табуретка со свойствами: Высота (h, в см), Качество изделия (низкое, среднее, высокое). Определить два виртуальных метода: «количество древесины», которое требует табуретка, по формуле 4·h+12, если качество низкое, и 5·h+14, если качество среднее или высокое, а также «стоимость», равная d·2, для низкого качества, d·3, для среднего качества, d·4, для высокого качества, где d – количество древесины, которое требует данный объект. Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Высоту, качество материала, количество древесины и стоимость. Создать также класс наследник Стул с дополнительным свойством: высота спинки (h2, в см), и переопределить метод «количество древесины», по формуле d+2h2+5, где d – количество древесины, которые требует табуретка с такими же параметрами (Метод «стоимость» не переопределять). В главной программе создать экземпляры классов Табуретка и Стул, и вывести информацию в таком виде: «табуретка» + информация о табуретке и «стул» + информация о стуле.

Программный код

#include <iostream>

using namespace std;

enum quality

{

low, middle, high

};

class Stool

{

public:

float h;

quality q;

virtual float amount\_wood()

{

if(q==low)

{

return 4\*h+12;

}

else

{

return 5\*h+14;

}

}

virtual float price()

{

float d = amount\_wood();

if(q==low)

{

return d\*2;

}

else if(q==middle)

{

return d\*3;

}

else

{

return d\*4;

}

}

string information()

{

string str="Height: " + to\_string(h) + "\nQuality: ";

switch(q)

{

case low: {str += "low";} break;

case middle: {str += "middle";} break;

case high: {str += "high";} break;

}

str += "\nQuantity: " + to\_string(amount\_wood()) + "\nPrice: " + to\_string(price());

return str;

}

};

class Chair : public Stool

{

public:

float spinka;

virtual float amount\_wood()

{

return Stool::amount\_wood()+2\*spinka+5;

}

};

int main()

{Stool first;

cout << "Enter the height of the stool: ";

cin >> first.h;

cout << "Product quality:\n 1.low;\n 2.middle;\n 3.high.\nChoose quality: ";

int n;

cin >> n;

switch(n)

{

case 1: {first.q = low;} break;

case 2: {first.q = middle;} break;

case 3: {first.q = high;} break;

}

cout << "Stool:\n" << first.information() << endl;

Chair second;

cout << "Enter the height of the chair: ";

cin >> second.h;

cout << "Product quality:\n 1.low;\n 2.middle;\n 3.high.\nChoose quality: ";

cin >> n;

switch(n)

{

case 1: {second.q = low;} break;

case 2: {second.q = middle;} break;

case 3: {second.q = high;} break;

}

cout << "Enter the height of spinka: ";

cin >> second.spinka;

cout << "Chair:\n" << second.information() << endl;

return 0;

}

### Результат

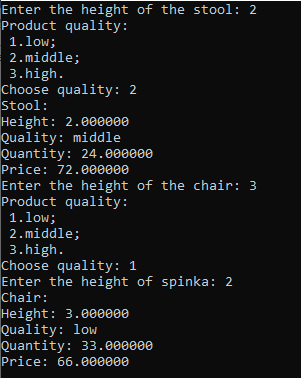


Рисунок 5 – Результат задания 1

## Контрольные вопросы

1. Какая функция называется виртуальной?

Виртуальная функция – это функция-член класса, объявленная в базовом классе и переопределенная в производном. Для ее объявления используется ключевое слово virtual.

1. Чем виртуальные функции отличаются от перегружаемых?
2. У перегруженных функций совпадают только имена, а количество и/или типы параметров должны отличаться, чтобы компилятор мог выбрать нужный вариант функции. У виртуальных функций же должны совпадать и имена, и количество параметров, и типы параметров.
3. Виртуальные функции не могут быть статическими членами класса.
4. Какой класс называется абстрактным?

Класс, содержащий хотя бы одну чисто виртуальную функцию, называется абстрактным.

1. В чем состоит различие раннего и позднего связывания?

Различие раннего и позднего связывания состоит непосредственно в том, когда компилятор принимает решение о вызове функции. При отсутствии виртуальных функций решение о вызове принимается компилятором на этапе компиляции программы(раннее связывание), а при наличии виртуальных функций компилятор откладывает решение до фактического запуска программы(позднее связывание).

1. Опишите назначение виртуального деструктора.

Назначение виртуального деструктора заключается в том, чтобы при вызове деструктора для производного класса не вызывался деструктор базового класса и не происходило удаление той части объекта, которая относится к базовому классу.

### Вывод

В результате лабораторной работы научились использовать виртуальные функции при наследовании.

Лабораторная работа № 6  
Тема: «МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ, ВИРТУАЛЬНОЕ НАСЛЕДОВАНИЕ»

### **Цель**

Изучить принципы и получить практические навыки при использовании множественного наследования; рассмотреть случаи, когда необходимо использовать виртуальное наследование.

## Задания

### Задание 1.6

### Условие

Первый базовый класс – Млекопитающие; поля – способ питания, вес, среда обитания; производные классы – хищники и травоядные. Во втором базовом классе описываются географические регионы.

Программный код

#include <iostream>

using namespace std;

enum food

{

grazing, predation, parasitism //выедание, хищничество, паразитизм

};

enum habitat

{

terrestrial, underground, aquatic, aerial, woody //наземные, подземные, водные, воздушные, древесные

};

enum continents

{

Eurasia, Africa, North\_America, South\_America, Australia, Antarctica

};

class Mammals

{protected:

int weight;

food a;

habitat b;

public:

Mammals(int weight, food a, habitat b)

{

this->weight = weight;

this->a = a;

this->b = b;

}

Mammals(){}

string GetVivod()

{

string str = "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\nWeight: " + to\_string(weight) + " kg\nFood: ";

switch(a)

{

case grazing: {str += "grazing";} break;

case predation: {str += "predation";} break;

case parasitism: {str += "parasitism";} break;

}

str += "\nHabitat: ";

switch(b)

{

case terrestrial: {str += "terrestrial";} break;

case underground: {str += "underground";} break;

case aquatic: {str += "aquatic";} break;

case aerial: {str += "aerial";} break;

case woody: {str += "woody";} break;

}

str += "\nContinents: ";

return str;

}

};

class Regions

{

protected:

continents c;

public:

Regions(continents c)

{

this->c = c;

}

Regions(){}

void GetC()

{

switch(c)

{

case Eurasia: cout << "Eurasia\n"; break;

case Africa: cout << "Africa\n"; break;

case North\_America: cout << "North America\n"; break;

case South\_America: cout << "South America\n"; break;

case Australia: cout << "Australia\n"; break;

case Antarctica: cout <<"Antarctica\n"; break;

}

}

};

class Predators : public Mammals, public Regions

{

public:

Predators(int weight, habitat b, continents c) : Mammals(), Regions()

{

this->weight = weight;

this->a = predation;

this->b = b;

this->c = c;

}

};

class Herbivores : public Mammals, public Regions

{

public:

Herbivores(int weight, habitat b, continents c) : Mammals(), Regions()

{

this->weight = weight;

this->a = grazing;

this->b = b;

this->c = c;

}

};

food function1(int a)

{

switch(a)

{

case 1: return grazing;

case 2: return predation;

case 3: return parasitism;

}

}

habitat function2(int b)

{

switch(b)

{

case 1: return terrestrial;

case 2: return underground;

case 3: return aquatic;

case 4: return aerial;

case 5: return woody;

}

}

continents function3(int c)

{

switch(c)

{

case 1: return Eurasia;

case 2: return Africa;

case 3: return North\_America;

case 4: return South\_America;

case 5: return Australia;

case 6: return Antarctica;

}

}

int main()

{

int a, b, c, weight;

cout << "Enter weight: ";

cin >> weight;

cout <<"\n1. Grazing\n2. Predation\n3. Parasitism\nEnter number: ";

cin >> a;

cout <<"\n1. Terrestrial\n2. Underground\n3. Aquatic\n4. Aerial\n5. Woody\nEnter number: ";

cin >> b;

cout <<"\n1. Eurasia\n2. Africa\n3. North America\n4. South America\n5. Australia\n6. Antarctica\nEnter number: ";

cin >> c;

Mammals first(weight, function1(a), function2(b));

Regions second(function3(c));

Predators third(weight, function2(b), function3(c));

Herbivores fourth(weight, function2(b), function3(c));

cout <<first.GetVivod();

second.GetC();

return 0;

}

### Результат



Рисунок 6 – Результат задания 1

## Контрольные вопросы

1. Для чего используется множественное наследование? Чем оно отличается от простого наследования?

Множественное наследование – это иерархическая структура, в которой производный класс наследует от нескольких базовых классов.

В случае простого наследования производный класс получает доступ к публичным и защищенным элементам базового класса.

В случае множественного наследования производный класс получает доступ к публичным и защищенным элементам всех базовых классов.

Кроме того, в момент создания экземпляра производного класса выполняться все конструкторы базовых классов, а при удалении объекта выполнятся все деструкторы базовых классов.

1. Каков механизм вызова конструкторов при множественном и виртуальном наследовании?

Множественное наследование позволяет порожденному классу наследовать элементы более, чем от одного базового класса.

Базовый класс может быть задан только один раз в списке порождения нового класса.

1. Какие проблемы возможны при множественном наследовании?

Первая – конфликт имен методов или атрибутов нескольких базовых классов.

Вторая возникает при многократном включении некоторого базового класса.

### Вывод

В результате лабораторной работы научились применять множественное наследование.

Лабораторная работа № 7  
Тема: «Шаблоны классов»

### **Цель**

Научиться использовать шаблоны функций для создания группы однотипных функций, а также шаблоны классов для создания группы связанных типов классов.

## Задания

### Задание 1.7

### Условие

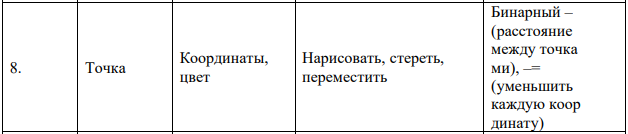


Рисунок 7 – Условие задания 1

Программный код

#include <iostream>

#include <cmath>

using namespace std;

template <typename T>

class Point

{private:

T coordinate, coordinate2;

string color;

public:

Point(T a=0, T b=0, string c = " ") : coordinate(a), coordinate2(b), color(c){}

void new\_coordinate(T x, T y)

{

this->coordinate = x;

this->coordinate2 = y;

}

void draw()

{

cout << coordinate << "\t" <<coordinate2 << "\t" << color<< endl;

}

void erase1()

{

this->coordinate = 0;

this->coordinate2 = 0;

this->color = " ";

}

void move1(Point &a)

{

cout << a.coordinate <<"\t" << a.coordinate2;

}

void vivod();

float operator - (Point &a)

{

float x = (sqrt((pow(this->coordinate - a.coordinate, 2) + pow(this->coordinate2 - a.coordinate2,2))));

return x;

}

Point operator -= (int x)

{

return Point(this->coordinate - x, this->coordinate2 - x);

}

Point operator += (int x)

{

return Point(this->coordinate + x, this->coordinate2 + x);

}

};

template <typename T>

T enter\_x()

{ T x;

cout << "Enter a coordinate(x): ";

cin >> x;

return x;

}

template <typename T>

T enter\_y()

{ T y;

cout << "Enter a coordinate(y): ";

cin >> y;

cout << "-----------------------------\n";

return y;

}

int main()

{ setlocale(LC\_ALL, "");

int x, y, z, key;

x = enter\_x<int>();

y = enter\_y<int>();

Point <int>first(x, y, "RED");

x = enter\_x<int>();

y = enter\_y<int>();

Point <int>second(x, y, "GREEN");

cout << "0. Exit\n1. Erase coordinates\n2. Move coordinates\n3. Distance between coordinates\n4. Draw:(\n5. Change coordinates\n";

do{

cout << "\nChoose a number: ";

cin >> key;

switch(key)

{

case 1: first.erase1(); second.erase1(); cout << "Coordinates erased!!!"<<endl; break;

case 2: {cout << "How much will we move the coordinates?: ";

cin >> z;

if(z>0)

{

first = (first +=z);

first.move1(first);

cout << endl;

second = (second +=z);

second.move1(second);

}

else

{

first = (first -=(-1\*z));

first.move1(first);

cout << endl;

second = (second -=(-1\*z));

second.move1(second);

}

break;

}

case 3: {float theed = second - first; cout << "Distance between coordinates:\n" << theed; } break;

case 4: first.draw(); second.draw(); cout <<"Drew:)"; break;

case 5: {x = enter\_x<int>(); y = enter\_y<int>(); first.new\_coordinate(x, y);

x = enter\_x<int>(); y = enter\_y<int>(); second.new\_coordinate(x, y);

}break;

}

}while(key!=0);

return 0;

}

### Результат

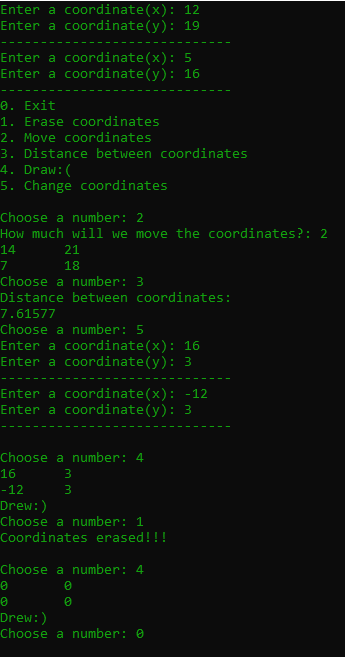


Рисунок 8 – Результат задания 1

## Контрольные вопросы

1. Для чего используются шаблоны классов? Что у них общего с шаблонами функций?

Шаблон семейства классов определяет способ построения отдельных классов подобно тому, как класс определяет правила построения и формат отдельных объектов. Этот класс можно рассматривать как некоторое описание множества классов, отличающихся только типами их данных. В С++ используется ключевое слово template для обеспечения параметрического полиморфизма. Шаблоны определения класса и шаблоны определения функции позволяют многократно использовать код, корректно по отношению к различным типам.

1. Как описываются шаблоны классов?

Формат шаблона класса имеет вид:

template <список параметров>

class объявление класса

1. Как создать объект на основе класса, порожденного шаблоном?

Так же, как и объект обычного класса, дополнительно указав параметризацию, например:

ИмяКласса <Параметризация> имяОбъекта (параметры конструктора);

1. Каких типов могут быть фактические параметры шаблонов классов?

Любых.

1. Можно ли описывать в списке параметров шаблона параметры, не определяющие тип?

Можно. Для этого в угловых скобках у нужного параметра вместо слова class пишется нужный тип данных.

### Вывод

В результате лабораторной работы научились работать с шаблонами классов.

Лабораторная работа № 8  
Тема: «Обработка исключений»

### **Цель**

Ознакомиться с понятием исключительных ситуаций и способами их обработки, изучить методы создания классов исключений и их объектов, способов формирования исключений в программе.

## Задания

### Задание 1.8

### Условие

Для каждого варианта необходимо создать три массива a, b и с размерами соответственно n1, n2 и n3 (n1≠n2≠n3). В массив a занести значения функции f(x) согласно варианту (при возникновении исключения заносить нули). Массив b заполнить случайными числами (среди них должны быть и отрицательные числа, и нули). Массив с формируется согласно варианту. Предусмотреть и обработать возникающие при этом исключительные ситуации (деление на ноль, корень из отрицательного числа, арифметическое переполнение, выход за пределы диапазона индексов массива и т.п.). Варианты заданий приведены в таблице.



Рисунок 9 – Условие задания 1

Программный код

#include <iostream>

#include <cmath>

#include <ctime>

using namespace std;

//класс исключения "деление на ноль"

class ZeroDivideException

{

};

//класс исключения "корень из отрицательного числа"

class SqrtException

{

};

//класс исключения "выход за пределы индексов массива"

class IndexException

{

};

//класс исключения "логарифм от отрицательного чилса"

class LogException

{};

int setN(int i)

{

int n;

cout << "Enter n" << i << ": ";

cin >> n;

return n;

}

int main()

{

int n1, n2, n3, i=0, MAX=10, MIN=-10;

double x=-3;

srand(time(NULL));

do

{

n1 = setN(1);

n2 = setN(2);

n3 = setN(3);

}while(n1 == n2 || n2 == n3 || n1 == n3 || n1<=0 || n2<=0 || n3<=0);

double \*a = new double[n1],

\*b = new double[n2],

\*c = new double[n3];

for(x=-3; x<=7; x+=0.5)

{

try

{

if(i >= n1)

{

throw IndexException();

}

if(x==-1)

{

throw ZeroDivideException();

}

double temp = (x-1)/(x+1);

if(temp<=0)

{

throw LogException();

}

a[i] = log(temp);

cout << "a[" << i << "] = " << a[i] << endl;

i++;

}

catch(IndexException)

{

cout << "Caught IndexException\n";

break;

}

catch(ZeroDivideException)

{

a[i] = 0;

cout << "a[" << i << "] = " << a[i] <<" - Caught ZeroDivideException"<<endl;

i++;

}

catch(LogException)

{

a[i] = 0;

cout << "a[" << i << "] = " << a[i] <<" - Caught LogException"<<endl;

i++;

}

}

for(i; i<n1; i++)

{

a[i] = 0;

cout << "a[" << i << "] = " << a[i] << endl;

}

cout << endl;

for(i=0; i<n2; i++)

{

b[i] = (float)rand()/RAND\_MAX \* (MAX-MIN)+(MIN);

if(b[i]>-1 && b[i]<1)

{

b[i] = 0;

}

cout << "b[" << i <<"] = " << b[i] << endl;

}

cout << endl;

for(i=0; i<n3; i++)

{

try

{

if(i>=n1)

{

throw IndexException();

}

if(i>=n2)

{

throw IndexException();

}

double temp = a[i]+b[i];

if(temp<0)

{

throw SqrtException();

}

c[i] = sqrt(temp);

cout << "c[" << i <<"] = " << c[i] <<endl;

}

catch(SqrtException)

{

c[i] = 0;

cout << "c[" << i <<"] = " << c[i] <<" - Caught SqrtException"<<endl;

}

catch(IndexException)

{

cout << "Caught IndexException" << endl;

break;

}

}

for(i; i<n3; i++)

{

c[i] = 0;

cout << "c[" << i <<"] = " << c[i] << endl;

}

delete []a;

delete []b;

delete []c;

return 0;

}

### Результат

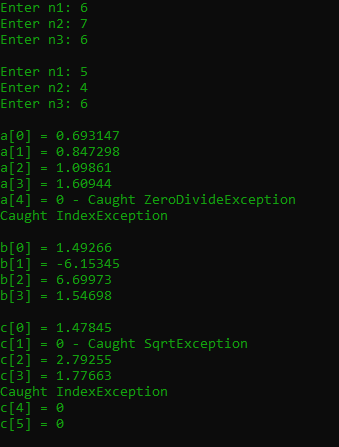


Рисунок 10 – Результат задания 1

## Контрольные вопросы

1. Что такое исключительная ситуация? Опишите ситуации, когда возникают исключения.

Исключительная ситуация – это возникновение непредвиденного или аварийного события, которое может порождаться некорректным использованием аппаратуры.

При выполнении операторов программы может возникнуть ошибочная ситуация (деление на ноль, корень из отрицательного числа, выход за пределы массива). Для обработки таких ситуаций можно предпринять одно из следующих действий: прервать выполнение программы; возвратить код ошибки; вывести сообщение об ошибке и вернуть программе некоторое значение, которое позволит ей продолжить работу. Для решения таких проблем и используются средства генерации и обработки исключений.

1. Как выделить в программе контролируемый блок?

Для выделения контролируемого блока в C++ используется служебная конструкция try{операторы}.

1. В каком месте кода программы может быть сгенерировано исключение?

Исключение может быть сгенерировано в любом месте программы, где возможно возникновение какого-либо аварийного или непредвиденного события, т. е. исключительной ситуации.

1. С помощью какого ключевого слова осуществляется генерация исключительных ситуаций? Укажите существующие формы этого оператора.

Генерация исключения осуществляется с помощью ключевого слова throw. Это слово может употребляться как с параметром, так и без него: throw [выражение].

1. Перечислите основные достоинства и недостатки блока catch(...) {}.

Этот блок предоставляет возможность обрабатывать все исключения, даже если тип переданного параметра неизвестен. Однако это же является и минусом использования этого блока: все исключения обрабатываются одинаково, и более детальная обработка этим блоком не предоставляется.

1. Что случится, если имеются несколько подходящих для этого типа обработчиков?

В этом случае исключение будет обработано в том обработчике, который перехватит его первым (т. е. в том, который расположен выше, ближе к блоку try{}, в котором рассматриваемое исключение было сгенерировано).

1. Каким образом осуществляется обработка исключений во вложенных контролируемых блоках?

Если в каком-либо контролируемом блоке было сгенерировано исключение, то поиск подходящего обработчика будет осуществляться среди тех catch-блоков, которые относятся непосредственно к этому контролируемому блоку. Если же подходящего обработчика не нашлось или исключение было обработано не полностью, то поиск продолжается среди обработчиков внешнего контролируемого блока, если таковой имеется.

1. Каким образом можно ограничить типы исключений, генерируемых функцией?

Это можно сделать с помощью спецификации исключений. В заголовке функции можно задать список типов исключений, которые она может прямо или косвенно порождать. Этот список приводится в конце заголовка и предваряется ключевым словом throw, например, void func(int a) throw (ex\_l, ex\_2). Такое объявление означает, что функция func может сгенерировать только исключения ex\_l, ex\_2 и исключения, являющиеся производными от этих типов.

### Вывод

Ознакомился с понятием исключительных ситуаций и способами их обработки, изучил методы создания классов исключений и их объектов, способов формирования исключений в программе.

Лабораторная работа № 9  
Тема: «Разработка приложений с использованием типа String.»

### Цель

Изучить теоретические принципы и получить практические навыки разработки программ в среде Visual Studio с использованием типа string из стандартной библиотеки шаблонов STL.

## Задания

### Задание 1.9

### Условие

Дан текст. Преобразовать его по следующему правилу: если нет символа ‘\*’, то оставить его без изменения, иначе заменить каждый символ, встречающийся после первого вхождения символа‘\*’, на символ ‘-’.

Программный код

#include <iostream>

#include <cstring>

#include <fstream>

using namespace std;

int main()

{ string line;

int x, i;

ifstream in("input.txt", ios\_base::in);

ofstream out("output.txt", ios\_base::out);

if(in.is\_open() && out.is\_open())

{

while (getline(in, line))

{

cout << line << endl;

for (i=0; i < line.length(); i++)

{

if(line[i]=='\*')

{

x = i;

break;

}

}

for(i=0; i <line.length(); i++)

{

if(i<=x)

{

cout << line[i];

out << line[i];

}

else if(line[i]==' ' && i>x)

{

cout << " ";

out << " ";

}

else

{

cout << "-";

out << "-";

}

}

}

in.close();

out.close();

cout << endl;

}

else

{

cout << "Error opening file!!!" << endl << endl;

}

return 0;

}

### Результат

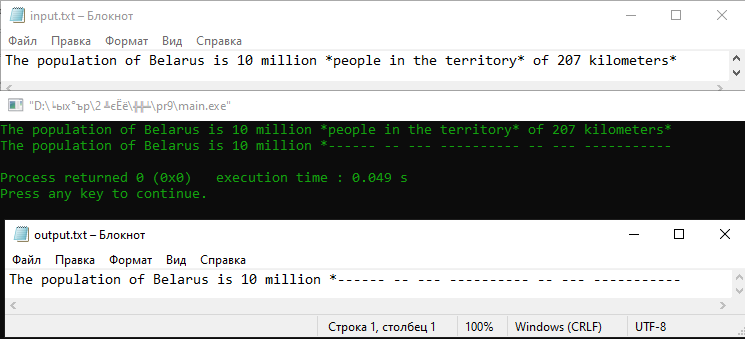


Рисунок 11 – Результат задания 1

## Контрольные вопросы

1. Какой класс используется для работы со строками в библиотеке STL? В каком заголовочном файле он определен?

string - это STL’евский класс основанный на шаблонах, который входит в стандартную библиотеку C++. Для использования данного класса в ваших приложениях, нужно подключить директиву <string>.

2. Перечислите основные виды конструкторов класса строк. Чем они отличаются?

*Конструкторы класса string:*

Конструктор1: string(const char \* s);

Описание1: Инициализирует объект класса string строкой с завершающим нулем (стандартная строка С) с указателем s.

Пример1: string one("Здравствуйте!");

Конструктор2: string(size\_type n, char c);

Описание2: Создает объект класса string размером n элементов, каждый из которых инициализируется символом с. (size\_type – это зависящий от реализации интегральный тип, определенный в строковом заголовочном файле.)

Пример2: string two(5, '$');

Конструктор3: string(const string \* str, size\_type pos = 0, size\_type n = npos);

Описание3: Инициализирует объект класса string объектом str, начиная с позиции pos в объекте str и завершая str, или копирует n симво лов, в зависимости от того, какое событие произойдет раньше. (npos – максимально возможная длина строки.)

Пример3: string three(one); string three1(one,10);

Конструктор4: string(const string \* str, size\_type n)

Описание4: Инициализирует объект класса string строкой с завершающим нулем (стандартная строка С) с указателем s и длиной n симво лов, даже если это превышает размер строки с завершаю

Пример4: char alls[] = "Ура!!! Каникулы!"; string five(alls,4);

Конструктор5: template<class Iter> string (Iter begin, Iter end)

Описание5: Инициализирует объект класса string значениями в диапазоне [begin, end), где begin и end действуют как указатели; диапазон включает в себя begin и простирается до end, но не включ

Пример5: string six(&five[7], &five[15]);

3. Чем отличается ввод/вывод стандартных строк С++ и строк библиотеки STL?

Отличается тем, что функция getline из библиотеки STL имеет разное количество параметров.

4. Назовите основные знаки операций работы со строками библиотеки STL. Какие действия они выполняют?

= (присваивает); + (сложение); += (добавляет в конец строки); >, <, <=, >=, ==, != (сравнение строк).

5. Перечислите основные функции работы со строками библиотеки STL. Для чего они предназначены?

Длину строки возвращает функция size() или length(); Функция max\_size()возвращает максимально допустимую длину строки. Метод at() обеспечивает такие же возможности доступа к отдельному символу строки, но номер используется как обозначение аргумента функции. Cпециальный метод empty() возвращает true для пустой строки и false для непустой. Строку также можно прибавить к другой с помощью метода append().Метод insert() позволяет вставлять строковый объект, массив, символ или несколько символов в строковый объект. Удалить все символы из строки позволяет функция clear(). Многочисленные методы replace() определяют часть строки, которую необходимо заменить, и саму замену. find() – поиск вхождения подстроки в строку.

### Вывод

Изучил теоретические принципы и получил практические навыки разработки программ в среде Visual Studio с использованием типа string из стандартной библиотеки шаблонов STL.