**Лабораторная работа 7**

**Программирование обработки исключительных ситуаций**

**Цель работы:** изучить синтаксис и семантику определения и вызова исключений, синтаксис обработчика и спецификации исключений; приобрести практические навыки запуска исключений; изучить особенности применения стандартных библиотечных исключений.

**Задание:**

Используя модифицированный АТД, обработайте все возможные исключительные ситуации.

**Код программы:**

#include <iostream>

#include <string>

#include <clocale>

using namespace std;

////////////////////////////////////////////////////////////////////////

//////////////////////////ПЕРВЫЙ БАЗОВЫЙ КЛАСС//////////////////////////

////////////////////////////////////////////////////////////////////////

class Meow1

{

protected:

int number;

int volume;

char\* color;

bool tempreg;

public:

Meow1();

Meow1(int a, int b, char\* c, bool d);

Meow1(const Meow1& a);

~Meow1();

Meow1 operator =(const Meow1 &a);

void Print() const;

Meow1 sum(const Meow1& a);

bool Cmp(const Meow1& a);

void Copy(const Meow1& a);

};

/////////////////////////////КОНСТРУКТОР ПО УМОЛЧАНИЮ/////////////////////////////

Meow1::Meow1()

{

number = 0;

volume = 0;

tempreg = false;

color = new char[1];

strcpy(color, "white");

}

/////////////////////////////КОНСТРУКТОР/////////////////////////////

Meow1::Meow1(int a, int b, char\* c, bool d)

{

if(a > 0) number = a;

else throw "Номер не может быть отрицательным. \n";

if(b > 0) volume = b;

else throw "Объем не может быть отрицательным. \n";

tempreg = d;

if (c != "") {

color = new char[strlen(c) + 1];

strcpy(color, c);

}

else

throw "Цвет не задан. \n";

}

/////////////////////////////КОПИРУЮЩИЙ КОНСТРУКТОР/////////////////////////////

Meow1::Meow1(const Meow1& a)

{

if (&a != this)

{

number = a.number;

volume = a.volume;

color = new char[strlen(a.color) + 1];

strcpy(color, a.color);

tempreg = a.tempreg;

}

else throw "Вы копируете не то и не туда. \n";

}

/////////////////////////////ДЕСТРУКТОР/////////////////////////////

Meow1 :: ~Meow1()

{

delete[] color;

}

/////////////////////////////ПЕРЕГРУЗКА ПРИСВАИВАНИЯ/////////////////////////////

Meow1 Meow1 :: operator=(const Meow1& a)

{

if (this == &a) return \*this;

else

{

this->Copy(a);

return \*this;

}

}

/////////////////////////////ФУНКЦИЯ ВЫВОДА/////////////////////////////

void Meow1::Print() const {

cout << "Серийный номер: " << number << endl;

cout << "Объём: " << volume << endl;

cout << "Регулятор температуры ";

if (tempreg)

{

cout << "имеется";

}

else

{

cout << "отсутствует";

}

cout << endl;

cout << "Цвет: " << color << endl;

};

/////////////////////////////ФУНКЦИЯ СЛОЖЕНИЯ/////////////////////////////

Meow1 Meow1::sum(const Meow1& a) {

Meow1 f;

char\* tmp = new char[strlen(color) + 1];

strcpy(tmp, color);

f.number = number + a.number;

f.volume = volume + a.volume;

f.tempreg = false;

strcat(tmp, "-");

strcat(tmp, a.color);

f.color = new char[strlen(tmp) + 1];

strcpy(f.color, tmp);

return f;

};

/////////////////////////////ФУНКЦИЯ СРАВНЕНИЯ/////////////////////////////

bool Meow1::Cmp(const Meow1& a) {

return volume == a.volume && number == a.number && tempreg == a.tempreg && color == a.color;

};

/////////////////////////////ФУНКЦИЯ КОПИРОВАНИЯ/////////////////////////////

void Meow1::Copy(const Meow1& a) {

number = a.number;

volume = a.volume;

color = new char[strlen(a.color) + 1];

strcpy(color, a.color);

tempreg = a.tempreg;

}

////////////////////////////////////////////////////////////////////////

//////////////////////////ВТОРОЙ БАЗОВЫЙ КЛАСС//////////////////////////

////////////////////////////////////////////////////////////////////////

class Meow3

{

protected:

int temper;

double length;

public:

Meow3();

Meow3(int a, double b);

Meow3(const Meow3& a);

Meow3 operator =(const Meow3 &a);

void Print() const;

void Switch();

};

/////////////////////////////КОНСТРУКТОР ПО УМОЛЧАНИЮ/////////////////////////////

Meow3::Meow3()

{

temper = 0;

length = 15;

}

/////////////////////////////КОНСТРУКТОР/////////////////////////////

Meow3::Meow3(int a, double b)

{

if(a > 0) temper = a;

else throw "Температура не может быть отрицательной. \n";

if(b > 0) length = b;

else throw "Длина не может быть отрицательной. \n";

}

/////////////////////////////КОПИРУЮЩИЙ КОНСТРУКТОР/////////////////////////////

Meow3::Meow3(const Meow3& a)

{

if (&a != this)

{

temper = a.temper;

length = a.length;

}

}

/////////////////////////////ПЕРЕГРУЗКА ПРИСВАИВАНИЯ/////////////////////////////

Meow3 Meow3 :: operator=(const Meow3& a)

{

temper = a.temper;

length = a.length;

return \*this;

}

/////////////////////////////ФУНКЦИЯ ВЫВОДА/////////////////////////////

void Meow3::Print() const {

cout << "Температура: " << temper << endl;

cout << "Длина прибора: " << length << endl;

};

/////////////////////////////ФУНКЦИЯ ВЫКЛЮЧЕНИЯ/////////////////////////////

void Meow3::Switch() {

temper = 0;

}

///////////////////////////////////////////////////////////////////////

///////////////////////////ПРОИЗВОДНЫЙ КЛАСС///////////////////////////

///////////////////////////////////////////////////////////////////////

class Meow2 : public Meow1, public Meow3

{

protected:

int size;

public:

Meow2();

Meow2(int a, int b, char\* c, bool d, int s, int t, double l);

Meow2(const Meow2& a);

~Meow2();

Meow2 operator =(const Meow2 &a);

void Print() const;

};

/////////////////////////////КОНСТРУКТОР ПО УМОЛЧАНИЮ/////////////////////////////

Meow2::Meow2() :Meow1(), Meow3()

{

size = 0;

temper = 0;

length = 0;

}

/////////////////////////////КОНСТРУКТОР/////////////////////////////

Meow2::Meow2(int a, int b, char\* c, bool d, int s, int t, double l) :Meow1(a, b, c, d), Meow3(t, l)

{

if(s > 0) size = s;

else throw "Размер не может быть отрицательным. \n";

temper = t;

length = l;

}

/////////////////////////////КОПИРУЮЩИЙ КОНСТРУКТОР/////////////////////////////

Meow2::Meow2(const Meow2& a)

{

if (&a != this) {

Meow1::Meow1(a);

Meow3::Meow3(a);

size = a.size;

temper = a.temper;

length = a.length;

}

}

/////////////////////////////ПЕРЕГРУЗКА ПРИСВАИВАНИЯ/////////////////////////////

Meow2 Meow2 :: operator=(const Meow2& a)

{

number = a.number;

volume = a.volume;

color = new char[strlen(a.color) + 1];

strcpy(color, a.color);

tempreg = a.tempreg;

size = a.size;

temper = a.temper;

length = a.length;

return \*this;

}

/////////////////////////////ДЕКСТРУКТОР/////////////////////////////

Meow2 :: ~Meow2()

{

Meow1::~Meow1();

}

/////////////////////////////ФУНКЦИЯ ВЫВОДА/////////////////////////////

void Meow2::Print() const

{

Meow1::Print();

Meow3::Print();

cout << "Размер нагревательного прибора: " << size << endl;

}

int main()

{

setlocale(LC\_ALL, "rus");

try {

Meow1 mur1(-1, 1, (char\*)"", true);

cout << "Информация о нагревательных элементах: " << endl;

cout << "/////////////////////////////////mur1/////////////////////////////////" << endl;

mur1.Print();

cout << endl;

}

catch(const char\* e) {

cout << e << endl;

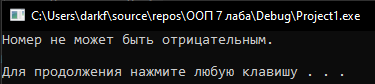
}

system("pause");

return 0;

}

**Результаты:**



**Контрольные вопросы:**

**1. Дайте определение исключения.**

Исключение – это возникающая в программе нештатная ситуация, с которой программа не может справиться. Например, при делении на ноль выполнение программы аварийно завершается системой. С++ дает возможность восстанавливать программу из ошибочных ситуаций и продолжать ее выполнение.

**2. В каком блоке описывается список обработчиков для возбужденного исключения?**

Список обработчиков для возбужденного исключения записывается в блоке catch.

**3. Опишите синтаксис обработчика исключения.**

Синтаксически обработчик имеет вид:

catch (формальный\_аргумент)

составная\_конструкция

**4. Что такое спецификация исключения?**

Синтаксически спецификация исключения является частью объявления и определения функции и имеет следующий вид: заголовок\_функции throw (список\_типов)

Здесь список\_типов – это список типов, которые может иметь выражение throw внутри функции.

**5. Для каких целей используется функция unexpected?**

Предоставляемая системой функция unexpected() вызывается, когда она возбудила исключение, которое отсутствует в ее списке спецификации исключений.