Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление: 710400 «Программная инженерия»

Дисциплина: « Объектно-ориентрованное проектирование / Объектно-ориентрованное программирование»

**ОТЧЕТ**

По лабораторной работе №9

Тема: «Исключения»

Выполнила: студентка группы

ПИ(б)-3-21 Пак Ксения

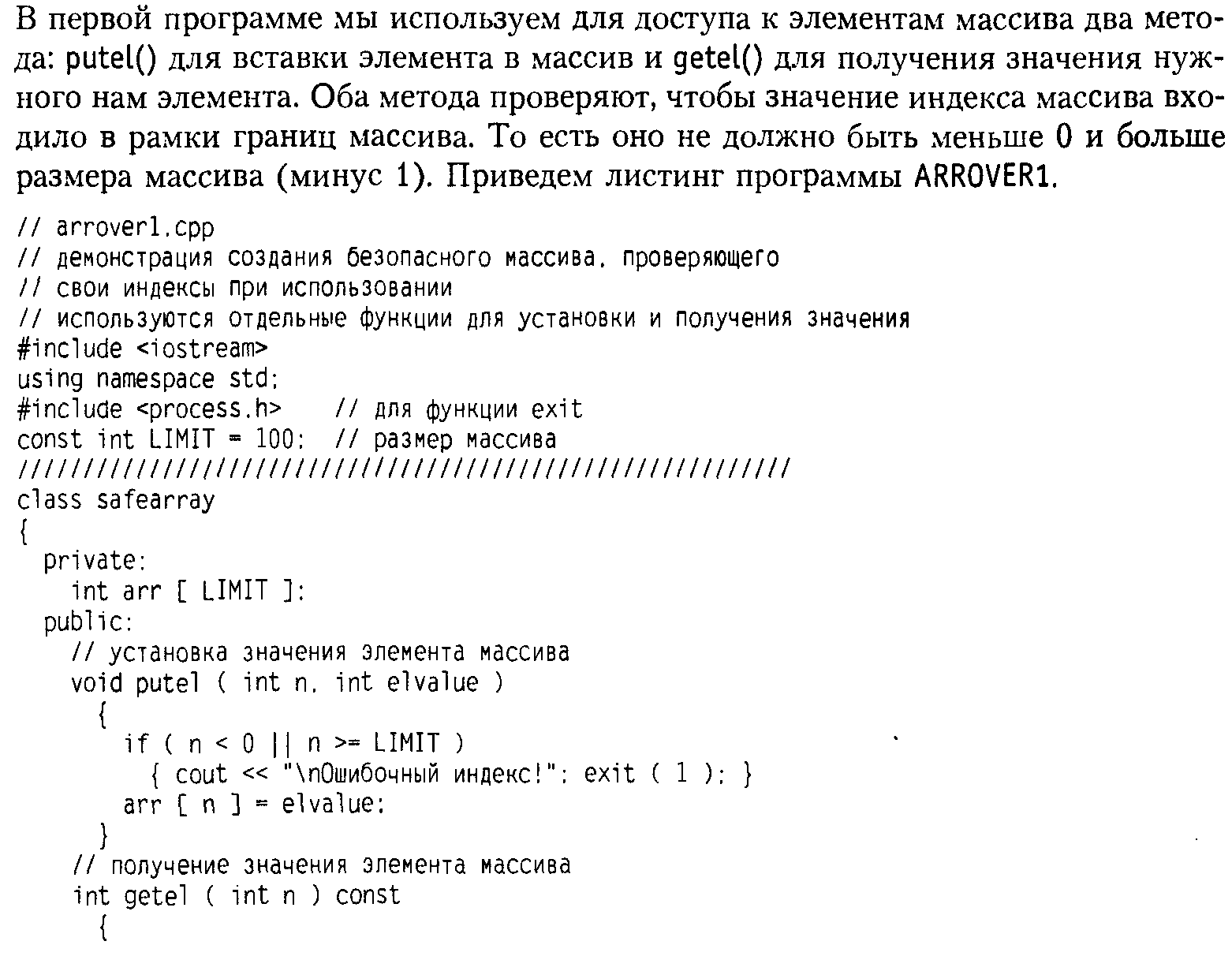
Проверил: Мусабаев Э. Б.

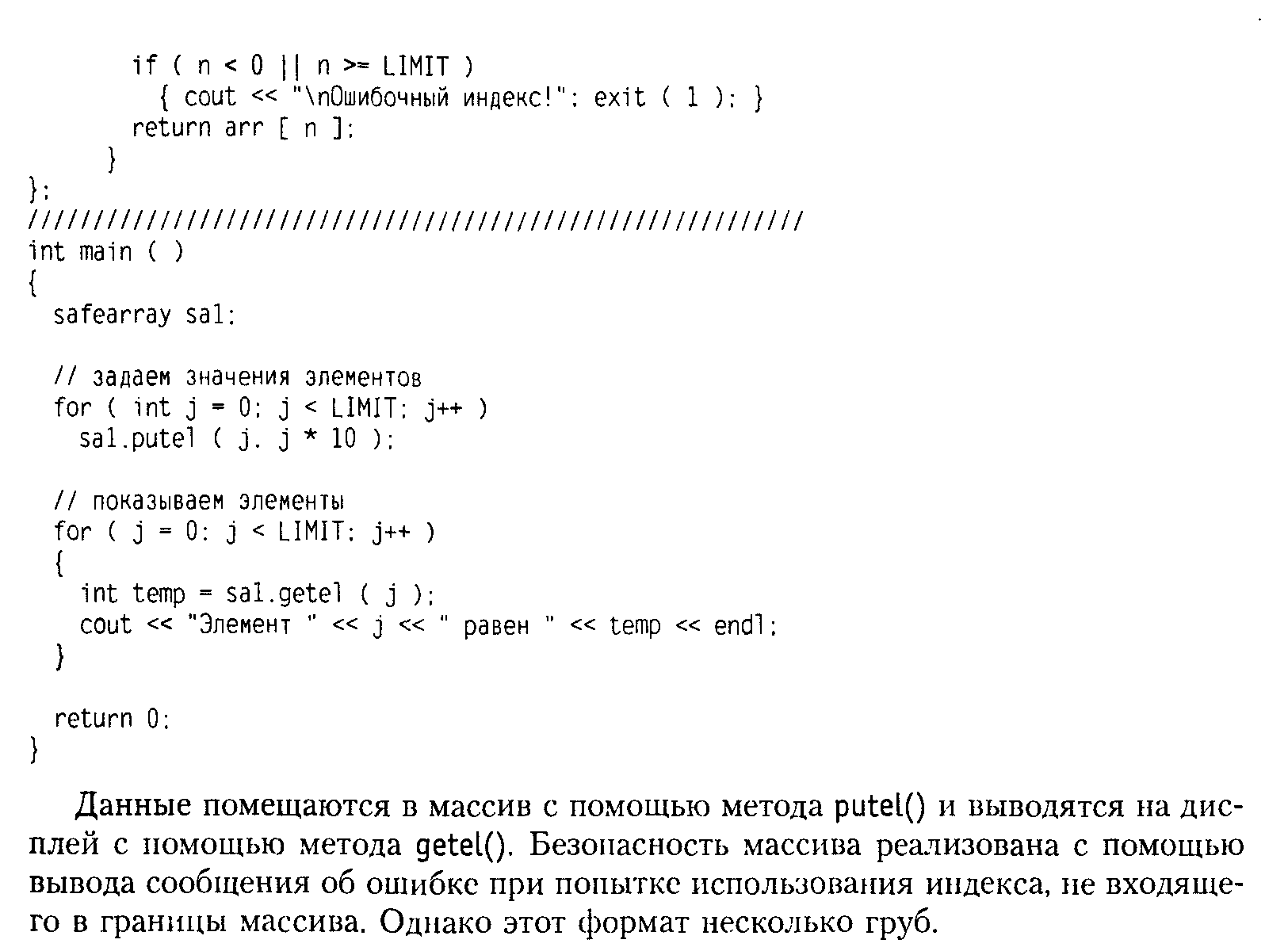
Бишкек – 2024

Задания

1. Добавьте класс исключений к программе ARROVER1, чтобы индексы, выходящие за пределы массива, вызывали генерацию исключения. Блок-ловушка может выводить пользователю сообщение об ошибке.

Листинг программы arrover1:





Данные помещаются в массив с помощью метода putel () и выводятся на экран с помощью метода getel (). Безопасность массива реализована с помощью вывода сообщений об ошибке при попытке использования индекса, не входящего в границы массива.

// arrover1.cpp

// демонстрация создания безопасного массива, проверяющего

// свои индексы при использовании

// используются отдельные функции для установки и получения значения

#include <Windows.h>

#include <iostream>

#include <process.h> // для функции exit

using namespace std;

const int LIMIT = 100; // размер массива

class safearray {

private:

int arr[LIMIT];

public:

class Range {};

// установка значения элемента массива

void putel(int n, int elvalue)

{

if (n < 0 || n >= LIMIT) {

throw Range();

}

arr[n] = elvalue;

}

// получение значения элемента массива

int getel(int n) const {

if (n < 0 || n > LIMIT) {

throw Range();

}

return arr[n];

}

};

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

safearray sal;

// исключение

try {

sal.putel(100, 10);

}

catch (safearray::Range) {

cout << "\nИСКЛЮЧЕНИЕ - ошибочный индекс!" << endl;

}

// задаем значения элементов

for (int j = 0; j < LIMIT; j++) {

try {

sal.putel(j, j \* 10);

}

catch (safearray::Range) {

cout << "\nИСКЛЮЧЕНИЕ - ошибочный индекс!" << endl;

}

}

// показываем элементы

for (int j = 0; j < LIMIT; j++) {

try {

int temp = sal.getel(j);

cout << "Элемент " << j << " равен " << temp << endl;

}

catch (safearray::Range) {

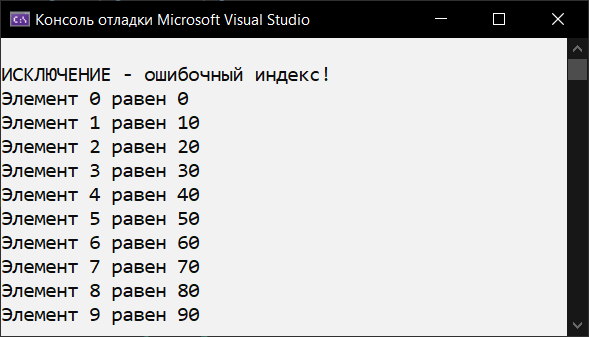
cout << "\nИСКЛЮЧЕНИЕ - ошибочный индекс!"<<endl;

}

}

return 0;

}



1. Задание 3 из лабораторной работы №4 выполнить с использованием механизма исключений.

#include <iostream>

#include <string>

#include <Windows.h>

class Tatra;

class SpeedExp {};

class Kamaz {

private:

std::string name;

double speed;

public:

Kamaz(const std::string& truckName, double truckSpeed) : name(truckName), speed(truckSpeed) {

if (speed < 0.0) {

throw SpeedExp();

}

}

void displayInfo() const {

std::cout << "Kamaz " << name << ": Speed - " << speed << " km/h\n";

}

double getSpeed() const {

return speed;

}

friend class Tatra;

};

class Tatra {

private:

std::string name;

double speed;

private:

double getSpeed() const {

return speed;

}

public:

Tatra(const std::string& truckName, double truckSpeed) : name(truckName), speed(truckSpeed) {

if (speed < 0.0) {

throw SpeedExp();

}

}

void displayInfo() const {

std::cout << "Tatra " << name << ": Speed - " << speed << " km/h\n";

}

friend int FrCreater(const Kamaz& kamaz, const Tatra& tatra);

};

int FrCreater(const Kamaz& kamaz, const Tatra& tatra) {

if (kamaz.getSpeed() > tatra.getSpeed()) {

return 1;

}

else if (kamaz.getSpeed() < tatra.getSpeed()) {

return -1;

}

else {

return 0;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

try {

Kamaz kamaz("Master", 80.0);

Tatra tatra("Phoenix", 75.0);

kamaz.displayInfo();

tatra.displayInfo();

int result = FrCreater(kamaz, tatra);

if (result == 1) {

std::cout << "Kamaz победил!\n";

}

else if (result == -1) {

std::cout << "Tatra победил!\n";

}

else {

std::cout << "Ничья! Скорости равны.\n";

}

}

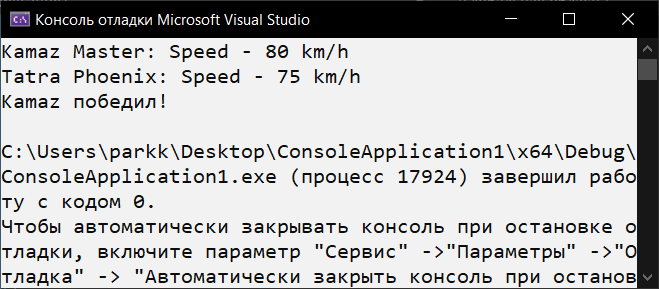
catch (SpeedExp) {

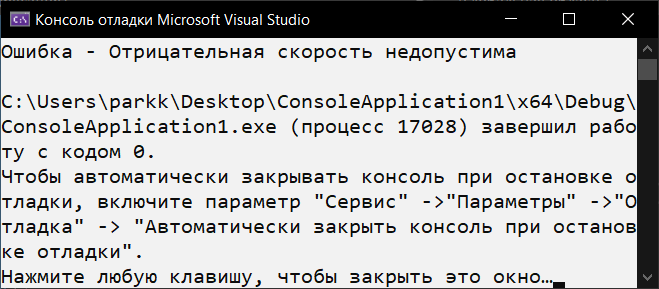
std::cerr << "Ошибка - Отрицательная скорость недопустима" << std::endl;

}

return 0;

}





В случае:

Kamaz kamaz("Master", -80.0);

Ответы на вопросы

1. Исключение в большинстве случаев возникает из-за:

* Программиста, написавшего исходный код приложения;
* Создателя класса, написавшего его метод;
* Ошибка выполнения; +
* Сбоя в операционной системе.

1. При работе с механизмом исключений в С++ используются следующие ключевые слова: \_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**catch, throw, try**

1. Исключения передаются:

* Из блока-ловушки в блок повторных попыток;
* Из выражения, создавшего исключительную ситуацию, в блок повторных попыток;
* Из точки, где возникла ошибка, в блок-ловушку. +
* Из выражения, в котором возникла ошибка, в блок-ловушку.

1. Для следующих ошибок обычно генерируются исключения:

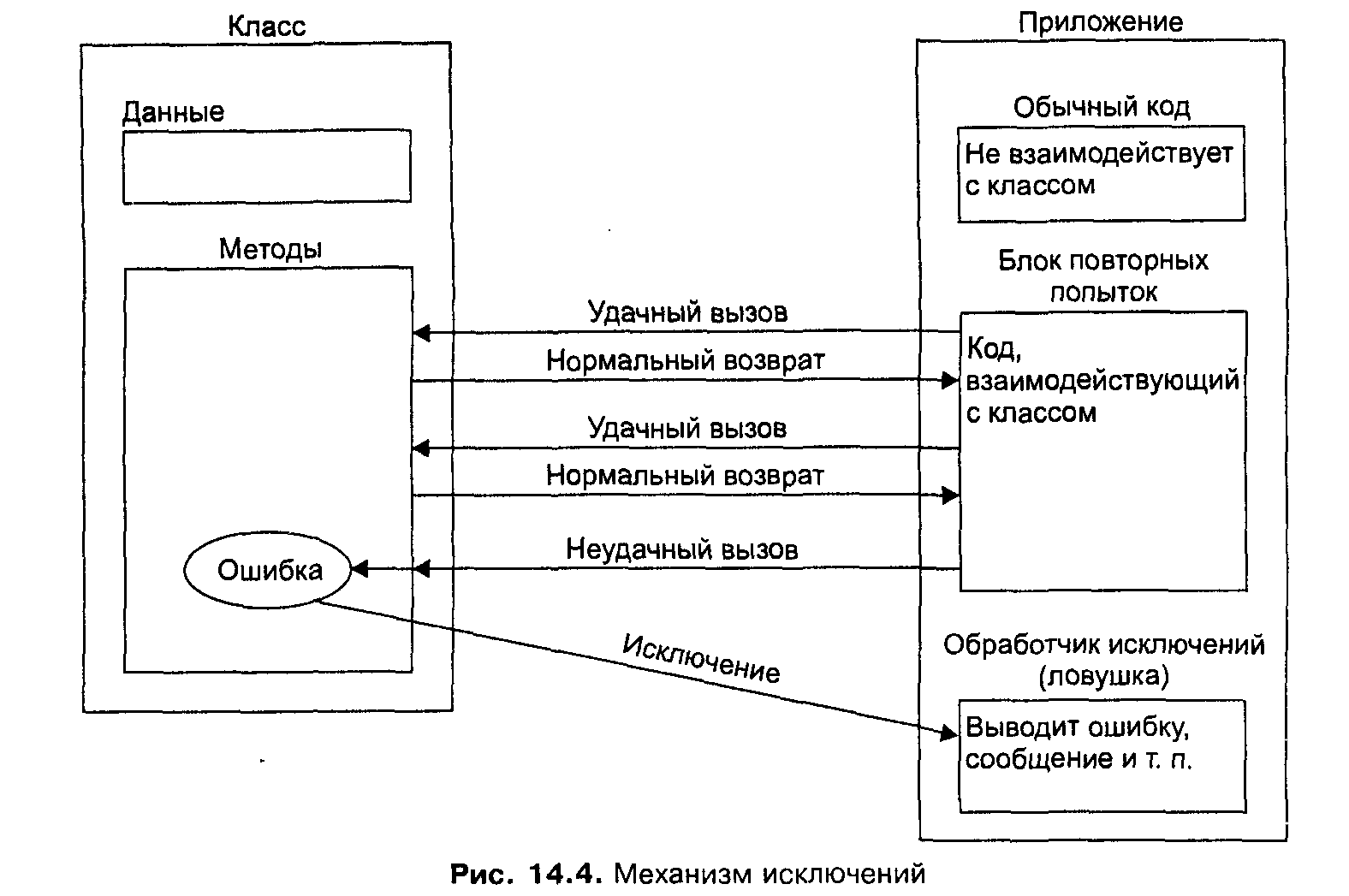
* Чрезмерное количество данных грозит переполнить массив; +
* Пользователь нажал Ctr+C для закрытия программы;
* Скачок напряжения в сети привел к перезагрузке системы;
* «New» не может зарезервировать необходимый объем памяти.

1. Истинно ли утверждение о том, что программа может продолжить свое выполнение после возникновения исключительной ситуации? Да.
2. Поясните механизм исключений.

*Генерация исключительной ситуации* – когда метод обнаруживает ошибку и информирует программу об этом.

*Обработчик исключительных ситуаций или улавливающий блок* – отдельная секция кода, в которой находится операции по обработке ошибок. В нем отлавливаются исключения, сгенерированные методами.

Любой код приложения, использующий объекты класса, заключается в *блок повторных попыток*. Ошибки, возникшие в *блоке повторных попыток*, будут пойманы *обработчиком исключений*. Код, который не имеет отношения к классу, не включается в блок повторных попыток.



Последовательность действий программы при возникновении ошибок:

1. Код нормально выполняется вне блока повторных попыток.
2. Управление переходит в блок повторных попыток.
3. Какое-то выражение в этом блоке приводит к возникновению ошибок в методе.
4. Метод генерирует исключение.
5. Управление переходит к обработчику ошибок (улавливающему блоку), следующему сразу за блоком повторных попыток.