Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа «Класс №9»**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

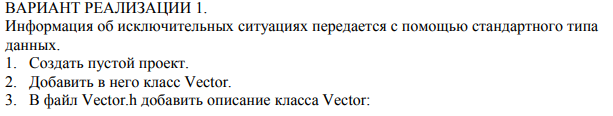
\_\_\_\_\_\_\_\_\_\_\_\_\_

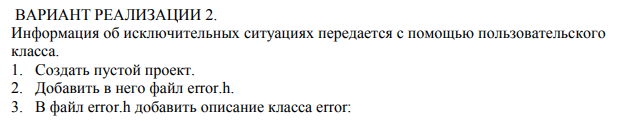
(дата)

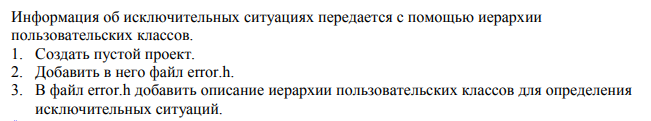
г.Пермь - 2022

**Постановка задачи**

1. Реализовать класс, перегрузить для него операции, указанные в варианте.
2. Определить исключительные ситуации.
3. Предусмотреть генерацию исключительных ситуаций







**Исходные данные**

Вариант 10:

Вариант реализации 1

#include <iostream>

using namespace std;

const int max\_size = 33;

class Variety

{

private:

int size;

int\* data;

public:

Variety();

Variety(int size);

//Variety(int s, int\* mas);

Variety(const Variety& v);

const Variety& operator=(const Variety& v);//операция присваивания

int& operator[](int i);

int& operator()();

Variety operator--();//удаление элемента

Variety operator-(const Variety& a);

//void ShowVariety();

friend ostream& operator<<(ostream& out, const Variety& v);

friend istream& operator>>(istream& in, Variety& v);

~Variety();

};

Variety::Variety()

{

size = 0;

data = 0;

}

Variety::Variety(int size)

{

if (size > max\_size) { throw 1; }

data = new int[size];

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

{

data[i] = rand() % 96;

}

}

/\*Variety::Variety(int s, int\* mas)

{

if (s > max\_size) throw error("Длина множества заходит за границы значения Max\_Size!");

size = s;

data = new int[size];

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

{

data[i] = mas[i];

}

}\*/

Variety::Variety(const Variety& v)

{

size = v.size;

data = new int[size];

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

{

data[i] = v.data[i];

}

}

/\*const Variety& Variety::operator=(const Variety& v)

{

if (this == &v)

{

return \*this;

}

size = v.size;

data = new int[size];

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

{

data[i] = v.data[i];

}

return \*this;

}

void Variety :: ShowVariety()

{

cout << "Variety: ";

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

{

cout << " " << data[i] << " ";

}

cout << endl;

}\*/

int& Variety :: operator[](int num)

{

if (num < size && num >= 0)

{

return data[num];

}

else

{

throw 2;

}

}

Variety Variety :: operator-(const Variety& a) //-a

{

if (size != a.size)

{

throw 3;

}

Variety tmp(size);

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

{

tmp.data[i] = data[i] - a.data[i];

}

delete[] data;

return tmp;

}

int& Variety :: operator()()

{

return size;

}

Variety Variety::operator--()

{

if (size == 0)//если элементов вообще нет

{

throw 4;

}

if (size == 1)//если в вектор один элемент

{

size = 0;

delete[] data;

data = 0;

return \*this;

};

int tmp\_size = size--;

Variety tmp(tmp\_size);

for (int i = 0; i < tmp\_size; i++)

{

tmp.data[i] = data[i];

}

delete[] data;

return tmp;

}

Variety :: ~Variety()

{

if (data != 0)

{

delete[] data;

}

}

ostream& operator<<(ostream& out, const Variety& v)

{

if (v.size == 0)

{

out << "Variety: NONE";

}

else

{

cout << "Variety: ";

for (int i = 0; i < v.size; i++)

{

out << v.data[i] << " ";

}

return out;

}

}

istream& operator>>(istream& in, Variety& v)

{

for (int i = 0; i < v.size; i++)

{

in >> v.data[i];

}

return in;

}

int main()

{

setlocale(LC\_ALL, "RUS");

/\*

const int max\_size = 33;

Действия:

[]-доступ по индексу

()-определение размера множества

- -разность множеств

-- -удаление элемента

\*/

try //контролирующий блок

{

Variety f(35);//Производит ошибку выхода size за Max\_Size

cout << f;

Variety a(5);

cout << a;

a[3];//Производит ошибку выхода поиска поиндексу за максимально возможный

Variety c(4);

cout << c;

Variety d = a - c;

cout << d;//Производит ошибку в разности множеств

Variety b(3);//пустое множество

cout << b;

--b;

--b;

--b;//Производит ошибку удаления элемента

}

catch (int n)

{

cout << "Номер ошибки: " << n << "....... " << endl << endl;

switch (n)

{

case 1:

cout << "Ошибка из-за выхода за границы MAX\_SIZE" << endl;

break;

case 2:

cout << "Ошибка действия доступа по индексу" << endl;

break;

case 3:

cout << "Ошибка в разности двух множеств" << endl;

case 4:

cout << "Ошибка действия по удалению элемента" << endl;

break;

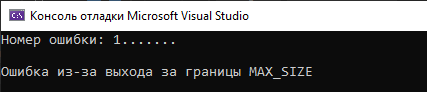
}

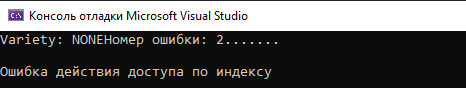
}

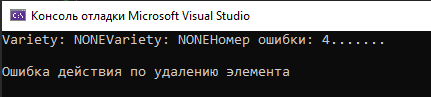
return 0;

}

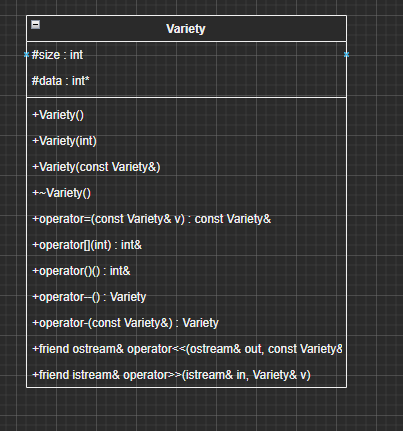
**Результат**







**UML – диаграмма**



Вариант реализации 2

#include <iostream>

#include <string>

using namespace std;

const int max\_size = 33;

class error //класс ошибка

{

private:

string sms;

public:

//конструктор инициализирует атрибут str смс об ошибке

error(string s)

{

sms = s;

}

void what()

{

cout << "Произошла ошибка: " << sms << endl; //выводит значение атрибута str

}

};

class Variety

{

private:

int size;

int\* data;

public:

Variety()

{

size = 0;

data = 0;

}

Variety(int s);

Variety(int s, int\* mas);

Variety(const Variety& v);

~Variety();

const Variety& operator=(const Variety& v);//операция присваивания

int& operator[](int i);

int& operator()();

Variety operator--();//удаление элемента

Variety operator-(const Variety& a);

//void ShowVariety();

friend ostream& operator<<(ostream& out, const Variety& v);

friend istream& operator>>(istream& in, Variety& v);

};

Variety::Variety(int s)

{

if (s > max\_size) throw error("Длина множества заходит за границы значения Max\_Size!");

data = new int[size];

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

{

data[i] = rand() % 96;

}

}

Variety::Variety(const Variety& v)

{

size = v.size;

data = new int[size];

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

{

data[i] = v.data[i];

}

}

Variety::~Variety()

{

if (data != 0) delete[] data;

}

/\*Variety::Variety(int s, int\* mas)

{

if (s > max\_size) throw error("Длина множества заходит за границы значения Max\_Size!");

size = s;

data = new int[size];

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

{

data[i] = mas[i];

}

}\*/

int& Variety::operator[](int i)

{

if (i < 0) throw error("Заданный индекс меньше нуля! ");

if (i >= size) throw error("Заданный индекс больше границы размера! ");

return data[i];

}

Variety Variety :: operator-(const Variety& a) //-a

{

if (size != a.size)

{

throw error("Ошибка в несоответствии кол-ва элементов для разности двух множеств!");

}

Variety tmp(size);

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

{

tmp.data[i] = data[i] - a.data[i];

}

delete[] data;

return tmp;

}

Variety Variety::operator--()

{

if (size == 0)//если элементов вообще нет

{

throw error ("Нечего удалять, так как множество нулевое!");

}

if (size == 1)//если в вектор один элемент

{

size = 0;

delete[] data;

data = 0;

return \*this;

};

int tmp\_size = size--;

Variety tmp(tmp\_size);

for (int i = 0; i < tmp\_size; i++)

{

tmp.data[i] = data[i];

}

delete[] data;

return tmp;

}

ostream& operator<<(ostream& out, const Variety& v)

{

if (v.size == 0)

{

out << "Variety: NONE";

}

else

{

cout << "Variety: ";

for (int i = 0; i < v.size; i++)

{

out << v.data[i] << " ";

}

return out;

}

}

istream& operator>>(istream& in, Variety& v)

{

for (int i = 0; i < v.size; i++)

{

cout << ">";

in >> v.data[i];

}

return in;

}

int main()

{

setlocale(LC\_ALL, "RUS");

/\*

const int max\_size = 33;

Действия:

[]-доступ по индексу

()-определение размера множества

- -разность множеств

-- -удаление элемента

\*/

try

{

//Variety f(35);//Производит ошибку выхода size за Max\_Size

//cout << f;

Variety a(5);

a[-1];//Производит ошибку выхода поиска поиндексу за максимально возможный

//Variety c(4);

//cout << c;

//Variety d = a - c;

//cout << d;//Производит ошибку в разности множеств

Variety b(3);//пустое множество

--b;

--b;

--b;//Производит ошибку удаления элемента

}

catch (error e)

{

e.what();

}

}

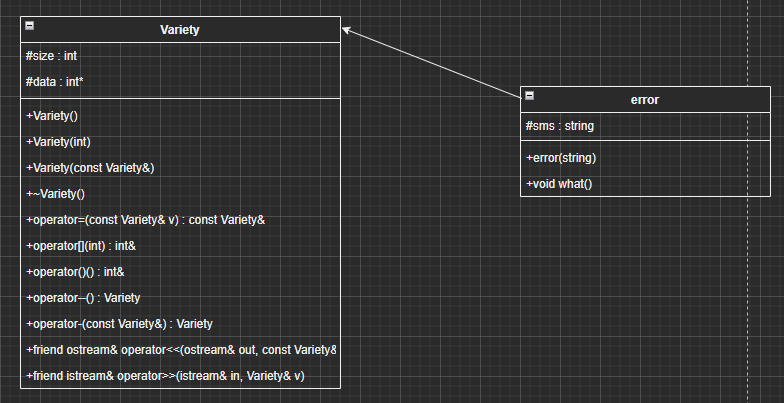
**Результат**  








**UML – диаграмма**



**Контрольные вопросы:**

* + 1. **Что представляет собой исключение в С++?**

Исключение – аварийное или непредвиденное событие.  
  
Исключение – это специальный объект класса или значение базового типа, который описывает (определяет) конкретную исключительную ситуацию и соответствующим образом обрабатывается.

* + 1. **На какие части исключения позволяют разделить вычислительный процесс? Достоинства такого подхода?**

Обнаружение и обработка. Удобно использовать в программе из нескольких модулей, не требуется возвращать значение в вызывающую функцию.

Исключения позволяют логически разделить вычислительный процесс на две части. — обнаружение аварийной ситуации и ее обработка. Это важно не только для лучшей структуризации программы. ... Раскручиванием стека называется процесс освобождения памяти из-под локальных переменных и возврата управления вызывающей функции. Когда функция завершается, происходит естественное раскручивание стека. Тот же самый механизм используется и при обработке исключений.

* + 1. **Какой оператор используется для генерации исключительной ситуации?**

Для их возбуждения используется оператор throw. Тип выражения, указанного в операторе throw, определяет тип исключительной ситуации, а значение может быть передано обработчику прерываний. Этот механизм, заявленный как стандартный, представляется весьма экзотическим без использования механизма классов.

* + 1. **Что представляет собой контролируемый блок? Для чего он нужен?**

Часть программы, которая будет выполняться до возникновения исключения.

* + 1. **Что представляет собой секция-ловушка? Для чего она нужна?**

Секция-ловушка с многоточием — это крайняя мера: если уж мы в нее попали, то в программе произошло что-то совсем непредусмотренное. Поэтому в такой секции обычно выводят сообщение о непредвиденном исключении и завершают работу программы.

* + 1. **Какие формы может иметь спецификация исключения в секции ловушке? В каких ситуациях используются эти формы?**

По значению, по ссылке, по константной ссылке, по указателю. Лучше всего передавать объект по ссылке, так как при этом не создается временный объект-исключение.

* + 1. **Какой стандартный класс можно использовать для создания собственной иерархии исключений?**

Exception  
  
Корневой класс исключения в стандартной иерархии называется exception. Он определен в стандартном заголовочном файле и является базовым для всех исключений, возбуждаемых функциями из стандартной библиотеки. Класс exception имеетследующийинтерфейс: namespace std. {class exception. public: exception() throw(); exception( const exception & ) throw()

* + 1. **Каким образом можно создать собственную иерархию исключений?**

При создании собственных исключений заканчивайте имя класса пользовательского исключения словом Exception и реализуйте три общих конструктора.

* + 1. **Если спецификация исключений имеет вид: void f1() throw (int,double); то какие исключения может прождать функция f1()?**

Int и double

* + 1. **Если спецификация исключений имеет вид: void f1()throw(); то какие исключенияможетпрождатьфункцияf1()?**

Любые

* + 1. **В какой части программы может генерироваться исключение?**

По идее, в любой, может встретиться в блоке try в самом начале, но обрабатывается catch в конце.

* + 1. **Написать функцию, которая вычисляет площадь треугольника по трем сторонам(формула Герона).**

Варианты реализации :

* + - * Без спецификации исключений;
      * со спецификацией throw();
      * с конкретной спецификацией с подходящим стандартным исключением;
      * спецификация с собственным реализованным исключением.