Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**Отчет по лабораторной работе “Классы” №9**

**по дисциплине**

**«Теория алгоритмов и структуры данных»**

Выполнил студент гр. ИВТ-21-1б

Ахунов Руслан Булатович

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Денис Владимирович

(оценка) (подпись)

г. Пермь - 2022

**Постановка задачи:**

1.  Реализовать класс, перегрузить для него операции, указанные в варианте.

2.  Определить исключительные ситуации.

3.  Предусмотреть генерацию исключительных ситуаций.

Вариант 2:

1 - Информация об исключительных ситуациях передается с помощью пользовательского класса.

Класс- контейнер ВЕКТОР с элементами типа int. Реализовать операции:

[]– доступа по индексу;

int() – определение размера вектора;

- n – удаляет n элементов из конца вектора;

+ n - добавляет n элементов в конец вектора.

2 - Информация об исключительных ситуациях передается с помощью иерархии пользовательских классов.

**Анализ задачи:**

class error

{

    string str;

public:

    error(string s) { str = s; }

    void what() { cout << str << endl; }

};

Класс ошибка, в котором есть конструктор, инициирует атрибут str сообщением об ошибке.

Метод what выводит значение атрибута str.

    Vector(int s)

    {

        if (s > MAX\_SIZE) throw error("Длина вектора больше макс. размера\n");

        size = s;

        beg = new int[s];

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

            beg[i] = 0;

    }

    Vector(int s, int\* mas)

    {

        if (s > MAX\_SIZE) throw error("Длина вектора больше макс. размера\n");

        size = s;

        beg = new int[s];

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

            beg[i] = mas[i];

    }

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

int operator[](int i)

    {

        if (i < 0) throw error("Индекс < 0\n");

        if (i >= size) throw error("Индекс > размера вектора\n");

        return beg[i];

    }

    Vector operator+(int a)

    {

        for (int i = 0; i < a; i++)

        {

            if (size + 1 >= MAX\_SIZE) throw error("Превышен макс. размер\n");

            size++;

        }

        Vector tmp(size, beg);

        for (int i = size - a; i < size; i++)

        {

            tmp.beg[i] = a;

        }

        return tmp;

    }

    Vector operator-(int a)

    {

        for (int i = 0; i < a; i++)

        {

            if (size - 1 < 0) throw error("Число убранных элементов больше, чем размер самого вектора");

            size--;

        }

Добавляем условия в различные методы, чтобы выводить ошибки.

class Error

{

public:

    virtual void what() {};

};

class IndexError : public Error

{

protected:

    string msg;

public:

    IndexError() { msg = "Ошибка индексации\n"; }

    virtual void what() { cout << msg; }

};

class SizeError : public Error

{

protected:

    string msg;

public:

    SizeError() { msg = "Ошибка размера вектора\n"; }

    virtual void what() { cout << msg; }

};

class MaxSizeError : public SizeError

{

protected:

    string msg\_M;

public:

    MaxSizeError() {

        SizeError();

        msg\_M = "Размер > MaxSize\n"; }

    virtual void what() { cout << msg << msg\_M; }

};

class EmptySizeError : public SizeError

{

protected:

    string msg\_E;

public:

    EmptySizeError() {

        SizeError();

        msg\_E = "Вектор пуст\n"; }

    virtual void what() { cout << msg << msg\_E; }

};

class IndexError0 : public IndexError

{

protected:

    string msg\_0;

public:

    IndexError0() {

        IndexError();

        msg\_0 = "Индекс < 0\n"; }

    virtual void what() { cout << msg << msg\_0; }

};

class IndexErrorS : public IndexError

{

protected:

    string msg\_S;

public:

    IndexErrorS() {

        IndexError();

        msg\_S = "Индекс > Размер вектора\n"; }

    virtual void what() { cout << msg << msg\_S; }

};

class IndexError\_S : public IndexError

{

protected:

    string msg\_S;

public:

    IndexError\_S() {

        IndexError();

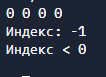
        msg\_S = "Индекс < Размер вектора\n"; }

    virtual void what() { cout << msg << msg\_S; }

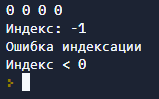
};

Вариант с помощью иерархии пользовательских классов, по своей сути, ничем не отличается, грубо говоря нужно заранее реализовать соответствующие производные классы, чтобы выводить ошибки

**Результат работы программы**

****

Через первый вариант



Через второй вариант

**Код программы**

**1 вариант**

#include <iostream>

#include <string>

using namespace std;

class error

{

    string str;

public:

    error(string s) { str = s; }

    void what() { cout << str << endl; }

};

const int MAX\_SIZE = 20;

class Vector

{

private:

    int size;

    int\* beg;

public:

    Vector()

    {

        size = 0;

        beg = 0;

    }

    Vector(int s)

    {

        if (s > MAX\_SIZE) throw error("Длина вектора больше макс. размера\n");

        size = s;

        beg = new int[s];

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

            beg[i] = 0;

    }

    Vector(int s, int\* mas)

    {

        if (s > MAX\_SIZE) throw error("Длина вектора больше макс. размера\n");

        size = s;

        beg = new int[s];

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

            beg[i] = mas[i];

    }

    Vector(const Vector& v)

    {

        size = v.size;

        beg = new int[size];

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

            beg[i] = v.beg[i];

    }

    ~Vector()

    {

        if (beg != 0) delete[]beg;

    }

    Vector& operator=(const Vector& v)

    {

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

        if (beg != 0) delete[]beg;

        size = v.size;

        beg = new int[size];

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

            beg[i] = v.beg[i];

        return \*this;

    }

    int operator[](int i)

    {

        if (i < 0) throw error("Индекс < 0\n");

        if (i >= size) throw error("Индекс > размера вектора\n");

        return beg[i];

    }

    Vector operator+(int a)

    {

        for (int i = 0; i < a; i++)

        {

            if (size + 1 >= MAX\_SIZE) throw error("Превышен макс. размер\n");

            size++;

        }

        Vector tmp(size, beg);

        for (int i = size - a; i < size; i++)

        {

            tmp.beg[i] = a;

        }

        return tmp;

    }

    Vector operator-(int a)

    {

        for (int i = 0; i < a; i++)

        {

            if (size - 1 < 0) throw error("Число убранных элементов больше, чем размер самого вектора");

            size--;

        }

        Vector tmp(size, beg);

        delete[]beg;

        beg = new int[size];

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

        {

            beg[i] = tmp.beg[i];

        }

        return \*this;

    }

    int operator()()

    {

        return size;

    }

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

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

};

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

{

    if (v.size == 0) out << "Пустой\n";

    else

    {

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

            out << v.beg[i] << " ";

        out << endl;

    }

    return out;

}

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

{

    for (int i = 0; i < v.size; i++)

    {

        cout << ">";

        in >> v.beg[i];

    }

    return in;

}

int main()

{

    setlocale(LC\_ALL, "Russian");

    try

    {

        Vector x(4);

        Vector y(1);

        cout << x;

        cout << "Индекс: ";

        int i;

        cin >> i;

        cout << x[i] << endl;

        cout << "Размер x: " << x() << endl;

        y = y + 3;

        cout << y;

        x = x - 3;

        cout << x;

        cout << "Размер x: " << x() << endl;

    }

    catch (error e)

    {

        e.what();

    }

    return 0;

}

**2 вариант:**

#include <iostream>

#include <string>

using namespace std;

class Error

{

public:

    virtual void what() {};

};

class IndexError : public Error

{

protected:

    string msg;

public:

    IndexError() { msg = "Ошибка индексации\n"; }

    virtual void what() { cout << msg; }

};

class SizeError : public Error

{

protected:

    string msg;

public:

    SizeError() { msg = "Ошибка размера вектора\n"; }

    virtual void what() { cout << msg; }

};

class MaxSizeError : public SizeError

{

protected:

    string msg\_M;

public:

    MaxSizeError() {

        SizeError();

        msg\_M = "Размер > MaxSize\n"; }

    virtual void what() { cout << msg << msg\_M; }

};

class EmptySizeError : public SizeError

{

protected:

    string msg\_E;

public:

    EmptySizeError() {

        SizeError();

        msg\_E = "Вектор пуст\n"; }

    virtual void what() { cout << msg << msg\_E; }

};

class IndexError0 : public IndexError

{

protected:

    string msg\_0;

public:

    IndexError0() {

        IndexError();

        msg\_0 = "Индекс < 0\n"; }

    virtual void what() { cout << msg << msg\_0; }

};

class IndexErrorS : public IndexError

{

protected:

    string msg\_S;

public:

    IndexErrorS() {

        IndexError();

        msg\_S = "Индекс > Размер вектора\n"; }

    virtual void what() { cout << msg << msg\_S; }

};

class IndexError\_S : public IndexError

{

protected:

    string msg\_S;

public:

    IndexError\_S() {

        IndexError();

        msg\_S = "Индекс < Размер вектора\n"; }

    virtual void what() { cout << msg << msg\_S; }

};

const int MAX\_SIZE = 20;

class Vector

{

private:

    int size;

    int\* beg;

public:

    Vector()

    {

        size = 0;

        beg = 0;

    }

    Vector(int s)

    {

        if (s > MAX\_SIZE) throw MaxSizeError();

        size = s;

        beg = new int[s];

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

            beg[i] = 0;

    }

    Vector(int s, int\* mas)

    {

        if (s > MAX\_SIZE) throw MaxSizeError();

        size = s;

        beg = new int[s];

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

            beg[i] = mas[i];

    }

    Vector(const Vector& v)

    {

        size = v.size;

        beg = new int[size];

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

            beg[i] = v.beg[i];

    }

    ~Vector()

    {

        if (beg != 0) delete[]beg;

    }

    Vector& operator=(const Vector& v)

    {

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

        if (beg != 0) delete[]beg;

        size = v.size;

        beg = new int[size];

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

            beg[i] = v.beg[i];

        return \*this;

    }

    int operator[](int i)

    {

        if (i < 0) throw IndexError0();

        if (i >= size) throw IndexErrorS();

        return beg[i];

    }

    Vector operator+(int a)

    {

        for (int i = 0; i < a; i++)

        {

            if (size + 1 >= MAX\_SIZE) throw MaxSizeError();

            size++;

        }

        Vector tmp(size, beg);

        for (int i = size - a; i < size; i++)

        {

            tmp.beg[i] = a;

        }

        return tmp;

    }

    Vector operator-(int a)

    {

        for (int i = 0; i < a; i++)

        {

            if (size - 1 < 0) throw IndexError\_S();

            size--;

        }

        Vector tmp(size, beg);

        delete[]beg;

        beg = new int[size];

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

        {

            beg[i] = tmp.beg[i];

        }

        return \*this;

    }

    int operator()()

    {

        return size;

    }

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

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

};

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

{

    if (v.size == 0) out << "Пустой\n";

    else

    {

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

            out << v.beg[i] << " ";

        out << endl;

    }

    return out;

}

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

{

    for (int i = 0; i < v.size; i++)

    {

        cout << ">";

        in >> v.beg[i];

    }

    return in;

}

int main()

{

    setlocale(LC\_ALL, "Russian");

    try

    {

        Vector x(4);

        Vector y(1);

        cout << x;

        cout << "Индекс: ";

        int i;

        cin >> i;

        cout << x[i] << endl;

        cout << "Размер x: " << x() << endl;

        y = y + 3;

        cout << y;

        x = x - 3;

        cout << x;

        cout << "Размер x: " << x() << endl;

    }

    catch (Error& e)

    {

        e.what();

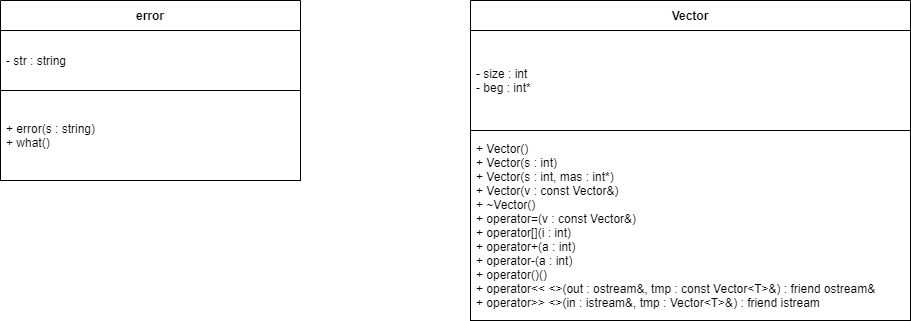
    }

    return 0;

}

**Диаграмма класса**

**1 вариант**



**2 вариант**

