ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е.О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «Обработка исключительных ситуаций» |
| по курсу: ОБЪЕКТНО ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134к |  |  |  | Костяков Н.А. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

**Цель работы**

Изучить принципы построения консольных приложений, применив на практике знания базовых синтаксических конструкций языка C++ и объектно-ориентированного программирования.

Общая часть задания: Создать класс Массив, содержащий: - одномерный целочисленный массив - размерность массива определяется суммой двух последних цифр шифра студента (если сумма получается меньше 12, то размерность массива будет равна 12) - поле, для хранения длины массива - конструктор, в котором осуществляется ввод данных в массив - расчетные функции (согласно варианта) - функцию вывода исходного массива - задействовать механизм обработки исключительных ситуаций (не менее 2-3 ситуаций)

**Вариант 7**

В одномерном массиве, состоящем из n целочисленных элементов, вычислить: 3 • номер максимального элемента массива; • произведение элементов массива, расположенных между первым и вторым нулевыми элементами. Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине — элементы, стоявшие в четных позициях.

**Листинг**

#include <iostream>

#include <string>

#include <math.h>

class NoZero

{

public:

    NoZero(const char\* msg = "Not neough zeros!!!")

    {

        text = msg;

    }

    const char\* what() { return text; }

private:

    const char\* text;

};

class arr

{

public:

    arr(int size);

    ~arr();

    void fill(int index);

    void show();

    int get\_size() { return size; }

    int\* get\_data() { return data; }

    int even\_multiplication(int\* arr, int size);

    int sum\_between\_zeros(int\* arr, int size);

    int\* convert(int \* arr, int size);

private:

    int\* data;

    int size;

};

arr::arr(int size)

{

    data = new int[size];

    this->size = size;

}

arr::~arr()

{

}

void arr::fill(int index)

{

    int a;

    std::cin>>a;

    data[index] = a;

}

void arr::show()

{

    for (int i = 0; i < size; i++)

    {

        std::cout << data[i] << " ";

    }

    std::cout << "\n";

}

int arr::even\_multiplication(int\* arr, int size)

{

    if (size<=1)

    {

        throw std::exception("Not Enough elements for even position multiplication\n");

    }

    int result = arr[0];

    for (int i = 2; i < size; i = i + 2)

    {

        result = result \* arr[i];

    }

    return result;

}

int arr::sum\_between\_zeros(int\* arr, int size)

{

        int zero1 = -1, zero2 = -1;

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

        {

            if (zero1 == -1 and arr[i] == 0)

            {

                zero1 = i;

                continue;

            }

            if (zero1 != -1 and arr[i] == 0)

            {

                zero2 = i;

                continue;

            }

        }

        //std::cout << "zero1: " << zero1 << " zero2: " << zero2 << '\n';

        if (zero2 == -1)

        {

            throw NoZero();

            return 0;

        }

        int result = 1;

        for (int i = zero1 + 1; i < zero2; i++)

        {

            result = result \* arr[i];

            //std::cout << result << '\n';

        }

        return result;

}

int\* arr::convert(int\* arr, int size)

{

    if (size<=1)

    {

        throw std::exception("Not enough elements for convertion\n");

    }

    int\* even = new int[0];

    int\* odd = new int[0];

    int e = 0;

    int o = 0;

    for (int i = 0; i < size; i++)

    {

        if (arr[i] % 2 == 0)

        {

            e++;

            int\* temp = even;

            even = new int[e];

            for (int q = 0; q < e-1; q++)

            {

                even[q] = temp[q];

            }

            even[e-1] = arr[i];

        }

        else

        {

            o++;

            int\* temp = odd;

            odd = new int[o];

            for (int q = 0; q < o - 1; q++)

            {

                odd[q] = temp[q];

            }

            odd[o-1] = arr[i];

        }

    }

    int\* result = new int[size];

    for (int i = 0; i < e; i++)

    {

        result[i] = even[i];

    }

    for (int i = e ; i < e+o; i++)

    {

        result[i] = odd[i-e];

    }

    return result;

}

int main()

{

    system("color f1");

    //out\_of\_range

    int size;

    std::cout << "Enter the size of array: ";

    std::cin >> size;

    arr list(size);

    std::cout << "Fill the array: ";

    for (int i = 0; i < size; i++)

    {

        try

        {

            list.fill(i);

        }

        catch (const std::exception& ex)

        {

            std::cout << ex.what();

        }

    }

    std::cout << "\nYour array: ";

    list.show();

    std::cout << "1) sum between zeros\n2) even pos mult\n3) sort the array\n";

    int var = 0;

    while (true)

    {

        std::cin >> var;

        switch (var)

        {

        case 1:

            try

            {

                int\* smth = list.get\_data();

                std::cout << list.sum\_between\_zeros(smth, size) << '\n';

            }

            catch (NoZero& ex)

            {

                std::cout << ex.what() << '\n';

            }

            break;

        case 2:

            try

            {

                std::cout<<"even mult: " << list.even\_multiplication(list.get\_data(), size)<<"\n";

            }

            catch (const std::exception& ex)

            {

                std::cout << ex.what() << '\n';

            }

            break;

        case 3:

            try

            {

                int\* res = list.convert(list.get\_data(), size);

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

                {

                    std::cout << res[i] << ' ';

                }

                std::cout << '\n';

            }

            catch (const std::exception& ex)

            {

                std::cout << ex.what()<<'\n';

            }

            break;

        default:

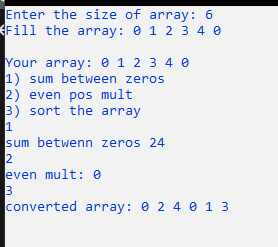
            break;

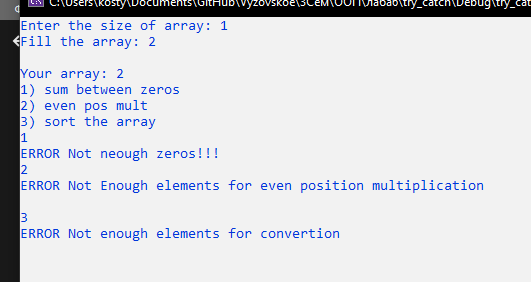
        }

    }

}

**Результат работы**





**Вывод**Я изучил и закрепил методы обработки исключительных ситуаций