**Федеральное агентство связи**

**Государственное бюджетное образовательное учреждение высшего**

**образование**

**Ордена Трудового Красного Знамени**

**«Московский технический университет связи и информатики»**

Кафедра информатики

**Отчет по лабораторной работе №6**

«Разработка программы обработки массива»

по дисциплине «Информатика»

Подготовил студент

группы БВТ1903: Щитов В.М.

Проверил: Волков А.И.

Москва 2020

Содержание

[1 Постановка задачи 3](#_Toc37348918)

[2 Схемы алгоритмов 5](#_Toc37348919)

[3 Текст программы 35](#_Toc37348920)

[4 Результаты тестирования программы 44](#_Toc37348921)

[Вывод 48](#_Toc37348922)

# **1 Постановка задачи**

Разработать на Visual C++ приложение «Обработка массива». Условие задачи предполагает следующие пункты его реализации:

1. Создать форму, имеющую вид, представленный на рисунке 1, приблизительный результат работы которой также представлен на рисунке 1.

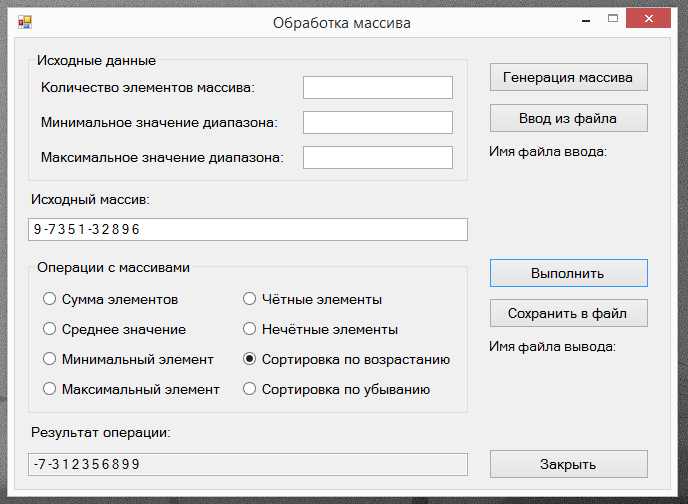


Рисунок 1 – Окно приложения «Обработка массива».

1. Программа должна корректно выполнять предусмотренные операции обработки введённого одномерного массива: определение суммы элементов массива, определение среднего значение элементов массива, определение значения минимального элемента массива, определение значения максимального элемента массива, вывод чётных значений элементов массива, вывод нечётных значений элементов массива, сортировка элементов массива по возрастанию, сортировка элементов массива по убыванию.
2. В программе должна быть предусмотрена возможность ввода одномерного массива целых чисел тремя способами: генерацией массива случайных чисел с заданными параметрами, вводом из существующего текстового файла или вводом исходного массива с клавиатуры в текстовое поле.
3. Для генерации массива случайных чисел должны использоваться три параметра: количество элементов массива, минимальное значение диапазона и максимальное значение диапазона.
4. В случае некорректных введённых данных для генерации массива должно выводиться сообщение об ошибке.
5. При вводе через файл строка массива должна передаваться в текстовое поле, где должны быть удалены все недопустимые для работы символы и их последовательности.
6. При вводе через текстовое поле реализовать обработчик очистки строки от некорректных символов и их последовательностей.
7. В программе должна быть реализована функция сохранения результатов, при этом в файл должны записываться такие данные, как: исходный массив, выполняемая с массивом операция и результат операции. При сохранении в файл должны избежать повторной записи уже внесённых в файл данных.
8. Программа должна корректно отображать данные, при возникновении ошибок, обрабатывать их.
9. При изменении исходных данных результаты расчёта должны «очищаться».
10. Нажатие на кнопку «Выход» должно корректно завершать работу программы.

# **2 Схемы алгоритмов**

На рисунке 2 изображена схема функции btnClose\_Click, используемой для закрытия программы. При этом перед закрытием программы появляется MessageBox, который дополнительно опрашивает пользователя о подтверждении закрытия программы.



Рисунок 2 – Схема алгоритма функции btnClose\_Click

На рисунке 3 изображена схема функции funcRestrictionOfInput, которая используется другими функциями для защиты ввода некорректных символов и их последовательностей в текстовых полях txtBoxInputNumberMas, txtBoxInputMinMas и txtBoxInputMaxMas.



Рисунок 3.1 – Схема алгоритма функции funcRestrictionOfInput



Рисунок 3.2 – Схема алгоритма функции funcRestrictionOfInput

На рисунке 4 изображена схема функции ChoiceOfAction, которая используется элементами типа RadioButton для изменения состояния переменной, которая используется для выбора выполняемой над массивом операции.



Рисунок 4 – Схема алгоритма функции ChoiceOfAction

На рисунках 5–7 изображены схемы алгоритмов функций txtBoxInputNumberMas\_TextChanged, txtBoxInputMinMas\_TextChanged и txtBoxInputMaxMas\_TextChanged соответственно. Все они вызывают ранее описанную функцию funcRestrictionOfInput с соответствующими аргументами.



Рисунок 5 – Схема алгоритма функции txtBoxInputNumberMas\_TextChanged



Рисунок 6 – Схема алгоритма функции

txtBoxInputMinMas\_TextChanged



Рисунок 7 – Схема алгоритма функции

txtBoxInputMaxMas\_TextChanged

На рисунке 8 изображена схема алгоритма функции txtBoxSourceMas\_TextChanged, которая отвечает за корректный ввод символов и их последовательностей в текстовое поле txtBoxSourceMas. Так как это после отличается от предыдущих текстовых полей тем, что в него вводятся элементы массива, то для него используется иная защита от некорректного ввода.



Рисунок 8 – Схема алгоритма функции

txtBoxSourceMas\_TextChanged

На рисунках 9–16 изображены схемы алгоритмов функций radioBtn\_CheckedChanged, которые используют функцию ChoiceOfAction. Данная функция выполняется при изменении активного RadioButton.



Рисунок 9 – Схема алгоритма функции radioBtn1\_CheckedChanged



Рисунок 10 – Схема алгоритма функции radioBtn2\_CheckedChanged



Рисунок 11 – Схема алгоритма функции radioBtn3\_CheckedChanged



Рисунок 12 – Схема алгоритма функции radioBtn4\_CheckedChanged



Рисунок 13 – Схема алгоритма функции radioBtn5\_CheckedChanged



Рисунок 14 – Схема алгоритма функции radioBtn6\_CheckedChanged



Рисунок 15 – Схема алгоритма функции radioBtn7\_CheckedChanged



Рисунок 16 – Схема алгоритма функции radioBtn8\_CheckedChanged

На рисунке 17 изображена схема функции StringCleanUp, используемой для очистки получаемой строки от недопустимых символов и лишних пробелов или знаков минуса.



Рисунок 17.1 – Схема алгоритма функции StringCleanUp



Рисунок 17.2 – Схема алгоритма функции StringCleanUp



Рисунок 17.3 – Схема алгоритма функции StringCleanUp

На рисунке 18 изображена схема функции StringSeparator, которая создаёт из массива строк одну строку. Данная функция используется при работе с файлами, из которых надо считать массив чисел.



Рисунок 18.1 – Схема алгоритма функции StringSeparator



Рисунок 18.2 – Схема алгоритма функции StringSeparator

На рисунке 19 изображена схема функции btnOpenFile\_Click, которая отвечает за открытие файла через нажатие на кнопку «Ввод из файла». Каждая строка файла считывается в массив строк, который передаётся в функцию StringSeparator, где каждая строка массива обрабатывается: в ней удаляются все недопустимые символы и их последовательности.



Рисунок 19.1 – Схема алгоритма функции btnOpenFile\_Click



Рисунок 19.2 – Схема алгоритма функции btnOpenFile\_Click

На рисунке 20 изображена схема функции btnGenerationMas\_Click, которая выводит массив случайных чисел в заданном диапазоне с заданным количеством элементов в текстовое поле txtBoxSourceMas.



Рисунок 20.1 – Схема алгоритма функции btnGenerationMas\_Click



Рисунок 20.2 – Схема алгоритма функции btnGenerationMas\_Click



Рисунок 20.3 – Схема алгоритма функции btnGenerationMas\_Click

На рисунке 21 изображена схема функции SaveFileFunc, которая используется при сохранении данных в файл, которая проверяет, какой исходный массив записывается в файл, и в случае совпадения записывает только наименование и результат проводимой с массивом операции.



Рисунок 21 – Схема алгоритма функции SaveFileFunc

На рисунке 22 изображена схема функции btnSaveFile\_Click, которая выполняется при нажатии на кнопку «Сохранить в файл». Данная функция использует функцию SaveFileFunc для проверки на наличие в файле уже сохранённого исходного массива или повторяющегося результата операции и самой повторяющейся операции.



Рисунок 22 – Схема алгоритма функции btnSaveFile\_Click

На рисунке 23 изображена схема функции funcConvertation, которая выполняется при любом действии и конвертирует данные из текстового поля txtBoxSourceMas в элементы массива.



Рисунок 23.1 – Схема алгоритма функции funcConvertation



Рисунок 23.2 – Схема алгоритма функции funcConvertation



Рисунок 23.3 – Схема алгоритма функции funcConvertation

На рисунке 24 изображена схема функции funcSumOfMas, которая отвечает за выполнение операции сложения элементов массива.



Рисунок 24.1 – Схема алгоритма функции funcSumOfMas



Рисунок 24.2 – Схема алгоритма функции funcSumOfMas

На рисунке 25 изображена схема функции funcAverage, которая отвечает за выполнение операции нахождения среднего элемента массива, при этом используя результат функции funcSumOfMas.



Рисунок 25 – Схема алгоритма функции funcAverage

На рисунке 26 изображена схема функции funcMinOfMas, которая отвечает за нахождение минимального элемента массива.



Рисунок 26 – Схема алгоритма функции funcMinOfMas

На рисунке 27 изображена схема функции funcMaxOfMas, которая отвечает за нахождение максимального элемента массива.



Рисунок 27 – Схема алгоритма функции funcMaxOfMas

На рисунке 28 изображена схема функции funcEvenNumbers, которая отвечает за вывод в TextBox всех чётных элементов массива.



Рисунок 28 – Схема алгоритма функции funcEvenNumbers

На рисунке 29 изображена схема функции funcUnevenNumbers, которая отвечает за вывод в TextBox всех нечётных элементов массива.



Рисунок 29 – Схема алгоритма функции funcUnevenNumbers

На рисунке 30 изображена схема функции funcSortOfMas1, которая отвечает за вывод отсортированного по возрастанию исходного массива. Для реализации сортировки был использован метод пузырьковой сортировки.



Рисунок 30.1 – Схема алгоритма функции funcSortOfMas1



Рисунок 30.2 – Схема алгоритма функции funcSortOfMas1

На рисунке 31 изображена схема функции funcSortOfMas2, которая отвечает за вывод отсортированного по убыванию исходного массива. Данная функция использует функцию funcSortOfMas1 и выводит полученный массив в обратном порядке.



Рисунок 31.1 – Схема алгоритма функции funcSortOfMas2



Рисунок 31.2 – Схема алгоритма функции funcSortOfMas2

На рисунке 32 изображена схема функции btnAction\_Click, которая выполняется при нажатии на кнопку «Выполнить». В дальнейшем в зависимости от выбранной через RadioButton операции, выполняется соответствующая функция.



Рисунок 32.1 – Схема алгоритма функции btnAction\_Click



Рисунок 32.2 – Схема алгоритма функции btnAction\_Click

# **3 Текст программы**

Исходный текст программы представлен в ниже приведённом листинге.

const size\_t MAX\_COUNT = 1024;

array<Int32>^ a = gcnew array<Int32>(MAX\_COUNT); //управляемый массив

int NumberOfElements = 0; //количество элементов массива

int TabIndexOfRadioButton = 0; //индекс переключателя

System::String^ ChoosenAction = "Сумма элементов"; //для вывода в файл выбранного действия

System::String^ Input = ""; //строка вводимого массива

System::String^ Result = ""; //строка результата

//Обработчик закрытия приложения

private: System::Void btnClose\_Click(System::Object^ sender, System::EventArgs^ e) {

System::Windows::Forms::DialogResult result = MessageBox::Show(this, "Вы уверены, что хотите выйти?", "Выход", MessageBoxButtons::YesNo, MessageBoxIcon::Exclamation, MessageBoxDefaultButton::Button2);

if (result == System::Windows::Forms::DialogResult::Yes) {

this->Close();

}

}

//Защита от дурака

private: System::Void funcRestrictionOfInput(System::Windows::Forms::TextBox^ object, System::Windows::Forms::KeyPressEventArgs^ input) {

if ((!Char::IsDigit(input->KeyChar)) && ((input->KeyChar != '-' || object->Text->IndexOf('-') != -1) && object != txtBoxSourceMas) && (input->KeyChar != ' ') && (input->KeyChar != (char)Keys::Back)) input->Handled = true;

if ((input->KeyChar == (char)Keys::D0 || input->KeyChar == (char)Keys::NumPad0) && (object->Text->Length >= 1) && (object->Text[0] == '0' && object->SelectionStart < 2)) input->Handled = true;

if (object == txtBoxInputNumberMas && (input->KeyChar == '-')) input->Handled = true; //количество не может быть отрицательным

if (object->Text->IndexOf('-') == 0) {

if (object->SelectionStart == 0) {

input->Handled = true;

}

if ((object->SelectionStart == 2) && (input->KeyChar == '0')) {

input->Handled = true;

}

}

if (input->KeyChar == '-' && object->Text->IndexOf('-') == -1) object->SelectionStart = 0;

else {

txtBoxResult->Text = "";

labelSaveFilePath->Text = L"Имя файла вывода:";

txtBoxSourceMas->Text = "";

}

}

//Функция выбора radioButton

private: System::Void ChoiceOfAction(System::Windows::Forms::RadioButton^ object){

int a = object->TabIndex;

if (a != TabIndexOfRadioButton) {

txtBoxResult->Text = "";

labelSaveFilePath->Text = L"Имя файла вывода:";

TabIndexOfRadioButton = a;

}

}

private: System::Void txtBoxInputNumberMas\_TextChanged(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

funcRestrictionOfInput(txtBoxInputNumberMas, e);

}

private: System::Void txtBoxInputMinMas\_TextChanged(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

funcRestrictionOfInput(txtBoxInputMinMas, e);

}

private: System::Void txtBoxInputMaxMas\_TextChanged(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

funcRestrictionOfInput(txtBoxInputMaxMas, e);

}

private: System::Void txtBoxSourceMas\_TextChanged(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

if ((!Char::IsDigit(e->KeyChar)) && (e->KeyChar != '-') && (e->KeyChar != ' ') && (e->KeyChar != (char)Keys::Back)) e->Handled = true;

else {

txtBoxResult->Text = "";

labelSaveFilePath->Text = L"Имя файла вывода:";

}

}

private: System::Void radioBtn1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

ChoiceOfAction(radioBtn1);

ChoosenAction = "Сумма элементов";

}

private: System::Void radioBtn2\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

ChoiceOfAction(radioBtn2);

ChoosenAction = "Среднее значение";

}

private: System::Void radioBtn3\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

ChoiceOfAction(radioBtn3);

ChoosenAction = "Минимальный элемент";

}

private: System::Void radioBtn4\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

ChoiceOfAction(radioBtn4);

ChoosenAction = "Максимальный элемент";

}

private: System::Void radioBtn5\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

ChoiceOfAction(radioBtn5);

ChoosenAction = "Чётные элементы";

}

private: System::Void radioBtn6\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

ChoiceOfAction(radioBtn6);

ChoosenAction = "Нечётные элементы";

}

private: System::Void radioBtn7\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

ChoiceOfAction(radioBtn7);

ChoosenAction = "Сортировка по возрастанию";

}

private: System::Void radioBtn8\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

ChoiceOfAction(radioBtn8);

ChoosenAction = "Сортировка по убыванию";

}

//Очистка строки входного файла

private: System::String^ StringCleanUp(System::String^ str) {

System::String^ str2 = "";

for (int i = 0; i < str->Length; i++) {

if (!Char::IsDigit(str[i]) && str[i] != ' ' && str[i] != '-') continue;

else {

if (i > 0 && i < (str->Length - 1)) {

if (str[i] == ' ' && str[i+1] == ' ') continue;

else if (str[i] == '-' && (Char::IsDigit(str[i-1]) && Char::IsDigit(str[i+1]))) str2 = str2 + " -";

else if (str[i] == '-' && ((str[i-1] == '-' || str[i+1] == '-') || str[i+1] == ' ' || Char::IsDigit(str[i-1]))) continue;

else str2 = str2 + str[i];

}

else if (i == 0 && str[i] == '-') {

if (str->Length > 2 && (str[i + 1] == ' ')) continue;

else str2 = str2 + str[i];

}

}

}

return str2;

}

//Для работы с массивами строк

private: System::String^ StringSeparator(array<String^>^ string) {

System::String^ str1 = "";

System::String^ str2 = "";

System::String^ str3 = "";

for (int i = 0; i < string->Length; i++) {

str1 = StringCleanUp(string [i]);

str2 = str2 + str1 + " ";

str1 = "";

}

for (int i = 0; i < str2->Length; i++) {

if (i > 0 && i < (str2->Length - 1)) {

if (str2[i] == ' ' && str2[i + 1] == ' ') continue;

else str3 = str3 + str2[i];

}

}

return str3;

}

//Диалог открытия файла, его считывание

private: System::Void btnOpenFile\_Click(System::Object^ sender, System::EventArgs^ e) {

OpenFileDialog^ OpenFileDlg = gcnew OpenFileDialog;

OpenFileDlg->Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

//фильтр открываемых файлов по форматам

OpenFileDlg->InitialDirectory = "E:\\"; //стартовая директория

if (OpenFileDlg->ShowDialog() == System::Windows::Forms:: DialogResult::OK)

{

labelOpenFilePath->Text = L"Имя файла ввода:";

System::String^ fileName = OpenFileDlg->FileName;

labelOpenFilePath->Text = labelOpenFilePath->Text + "\n" + fileName;

array<String^>^ str = gcnew array<String^>(100);

str = System::IO::File::ReadAllLines(fileName, System::Text::Encoding::UTF8);

String^ fileText = StringSeparator(str); //обработка текста

txtBoxSourceMas->Text = fileText;

}

}

//Обработчик генерации массива

private: System::Void btnGenerationMas\_Click(System::Object^ sender, System::EventArgs^ e) {

if (txtBoxInputNumberMas->Text == "" || txtBoxInputMaxMas->Text == "" || txtBoxInputMinMas->Text == "" || txtBoxInputMaxMas->Text == "-" || txtBoxInputMinMas->Text == "-") {

MessageBox::Show(this, "Введите числа!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

else if ((Convert::ToInt32(txtBoxInputNumberMas->Text)) == 0) {

MessageBox::Show(this, "Введите корректное число элементов элемента!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

else if ((Convert::ToInt32(txtBoxInputMinMas->Text)) > (Convert::ToInt32(txtBoxInputMaxMas->Text))) {

MessageBox::Show(this, "Минимальное значение массива больше максимального значения!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

else if ((Convert::ToInt32(txtBoxInputMinMas->Text)) == (Convert::ToInt32(txtBoxInputMaxMas->Text))) {

MessageBox::Show(this, "Минимальное и максимальное значения массива равны!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

else {

int i;

srand(time(0)); //рандомизация по времени

txtBoxSourceMas->Text = L"";

NumberOfElements = Convert::ToInt32(txtBoxInputNumberMas->Text);

for (i = 0; i < NumberOfElements; i++) {

Random^ temp = gcnew Random(i + rand());

a[i] = temp->Next((Convert::ToInt32(txtBoxInputMinMas->Text)), (Convert::ToInt32(txtBoxInputMaxMas->Text)+1));

txtBoxSourceMas->Text = txtBoxSourceMas->Text + " " + Convert::ToString(a[i]);

}

txtBoxInputNumberMas->Text = "";

txtBoxInputMinMas->Text = "";

txtBoxInputMaxMas->Text = "";

}

}

//Функция для сохранения файла (проверка массива)

private: System::Void SaveFileFunc(String^ file) {

int i;

const int size = 3;

array<String^>^ str = gcnew array<String^>(size); //массив строк

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

str[i] = "";

}

str[0] = "Исходный массив: " + txtBoxSourceMas->Text;

str[1] = "Действие: " + ChoosenAction;

str[2] = "Результат: " + txtBoxResult->Text;

String^ str1 = txtBoxSourceMas->Text;

String^ str2 = txtBoxResult->Text;

if (str1 == "" || str2 == "") {

MessageBox::Show(this, "Данные для сохранения отсутствуют!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Error);

}

else {

if (!System::IO::File::Exists(file)) {

System::IO::File::WriteAllLines(file, str); //создание и открытие файла

}

else {

if (str1 != Input){

System::IO::File::AppendAllText(file, "\r\n" + str[0] + "\r\n");

Input = str1;

}

if (str2 != Result) {

System::IO::File::AppendAllText(file, str[1] + "\r\n");

System::IO::File::AppendAllText(file, str[2] + "\r\n");

Result = str2;

}

}

}

}

//Диалог сохранения файла

private: System::Void btnSaveFile\_Click(System::Object^ sender, System::EventArgs^ e) {

SaveFileDialog^ SaveFileDlg = gcnew SaveFileDialog;

SaveFileDlg->Filter = "Text files(\*.txt)|\*.txt|All files(\*.\*)|\*.\*";

SaveFileDlg->InitialDirectory = "E:\\";

SaveFileDlg->OverwritePrompt = false;

if (SaveFileDlg->ShowDialog() == System::Windows::Forms:: DialogResult::OK)

{

labelSaveFilePath->Text = L"Имя файла вывода:";

System::String^ fileName = SaveFileDlg->FileName;

labelSaveFilePath->Text = labelSaveFilePath->Text + "\n" + fileName;

SaveFileFunc(fileName);

}

}

//Функция ввода массива через пробел (текстбокс)

private: System::Void funcConvertation(System::String^ str) {

int i, counter = 0;

System::String^ str1 = StringCleanUp(str) + str[str->Length - 1];

System::String^ tempStr = "";

for (i = 0; i < (str1->Length); i++) {

if ((!Char::IsDigit(str1[i]) && str1[i] != ' ' && str1[i] != '-') || (str1[i] == ' ' && i == 0)) continue;

if (Char::IsDigit(str1[i]) || str1[i] == '-') tempStr = tempStr + str1[i];

if (str1[i] == ' ' && tempStr != "" && tempStr != "-") {

a[counter] = Convert::ToInt32(tempStr);

tempStr = "";

counter = counter + 1;

}

if (i == (str1->Length - 1) && tempStr != "" && tempStr != "-") {

if (tempStr[tempStr->Length - 1] == '-') tempStr = tempStr->Remove(tempStr->Length - 1, 1);

a[counter] = Convert::ToInt32(tempStr);

tempStr = "";

counter = counter + 1;

}

}

NumberOfElements = counter;

txtBoxSourceMas->Text = "";

for (i = 0; i < NumberOfElements; i++) {

txtBoxSourceMas->Text = txtBoxSourceMas->Text + " " + Convert::ToString(a[i]);

}

}

//Функция суммы элементов массива

System::Void funcSumOfMas(System::Windows::Forms::TextBox^ TxtBox, array<Int32>^ arr) {

long long sum = 0;

int i;

for (i = 0; i < NumberOfElements; i++) {

sum = sum + arr[i];

}

TxtBox->Text = Convert::ToString(sum);

}

//Функция среднего элементов

System::Void funcAverage(System::Windows::Forms::TextBox^ TxtBox, array<Int32>^ arr){

funcSumOfMas(TxtBox, arr);

double sum = Convert::ToInt32(TxtBox->Text);

double result = sum / NumberOfElements;

TxtBox->Text = Convert::ToString(result);

}

//Функция нахождения минимального элемента массива

System::Void funcMinOfMas(System::Windows::Forms::TextBox^ TxtBox, array<Int32>^ arr) {

int minOfMas = 1000000000;

for (int i = 0; i < NumberOfElements; i++) {

if (arr[i] < minOfMas)minOfMas = arr[i];

}

TxtBox->Text = Convert::ToString(minOfMas);

}

//Функция нахождения максимального элемента массива

System::Void funcMaxOfMas(System::Windows::Forms::TextBox^ TxtBox, array<Int32>^ arr) {

int maxOfMas = -1000000000;

for (int i = 0; i < NumberOfElements; i++) {

if (arr[i] > maxOfMas) maxOfMas = arr[i];

}

TxtBox->Text = Convert::ToString(maxOfMas);

}

//Функция вывода чётных чисел

System::Void funcEvenNumbers(System::Windows::Forms::TextBox^ TxtBox, array<Int32>^ arr){

TxtBox->Text = "";

for (int i = 0; i < NumberOfElements; i++) {

if (arr[i] % 2 == 0) {

TxtBox->Text = TxtBox->Text + " " + Convert::ToString(arr[i]);

}

}

}

//Функция вывода нечётных чисел

System::Void funcUnevenNumbers(System::Windows::Forms::TextBox^ TxtBox, array<Int32>^ arr){

TxtBox->Text = "";

for (int i = 0; i < NumberOfElements; i++) {

if (arr[i] % 2 != 0) {

TxtBox->Text = TxtBox->Text + " " + Convert::ToString(arr[i]);

}

}

}

//Сортировка по возрастанию

System::Void funcSortOfMas1(System::Windows::Forms::TextBox^ TxtBox, array<Int32>^ arr){

int a, b, i;

TxtBox->Text = "";

// Реализация метода пузырьковой сортировки.

for (a = 1; a < NumberOfElements; a++)

for (b = NumberOfElements - 1; b >= a; b--) {

if (arr[b - 1] > arr[b]) { //Элементы неупорядочены.

//Меняем элементы местами.

i = arr[b - 1];

arr[b - 1] = arr[b];

arr[b] = i;

}

}

//Конец пузырьковой сортировки.

for (i = 0; i < NumberOfElements; i++) {

TxtBox->Text = TxtBox->Text + " " + Convert::ToString(arr[i]); //вывод

}

}

//Сортировка по убыванию

System::Void funcSortOfMas2(System::Windows::Forms::TextBox^ TxtBox, array<Int32>^ arr){

funcSortOfMas1(TxtBox, arr);

TxtBox->Text = "";

for (int i = NumberOfElements - 1; i >= 0; i--) {

TxtBox->Text = TxtBox->Text + " " + Convert::ToString(arr[i]);

}

}

//Выбор действия после нажатия кнопки

private: System::Void btnAction\_Click(System::Object^ sender, System::EventArgs^ e) {

if (txtBoxSourceMas->Text == L"") {

System::Windows::Forms::DialogResult result = MessageBox::Show(this, "Введите элементы массива!", "Ошибка!",MessageBoxButtons::OK, MessageBoxIcon::Error, MessageBoxDefaultButton::Button1);

}

else {

funcConvertation(txtBoxSourceMas->Text);

switch (TabIndexOfRadioButton) {

case 0:

funcSumOfMas(txtBoxResult, a);

break;

case 1:

funcAverage(txtBoxResult, a);

break;

case 2:

funcMinOfMas(txtBoxResult, a);

break;

case 3:

funcMaxOfMas(txtBoxResult, a);

break;

case 4:

funcEvenNumbers(txtBoxResult, a);

break;

case 5:

funcUnevenNumbers(txtBoxResult, a);

break;

case 6:

funcSortOfMas1(txtBoxResult, a);

break;

case 7:

funcSortOfMas2(txtBoxResult, a);

break;

}

}

}

# **4 Результаты тестирования программы**

На рисунке 33 представлен интерфейс рабочей программы и её корректная работа.

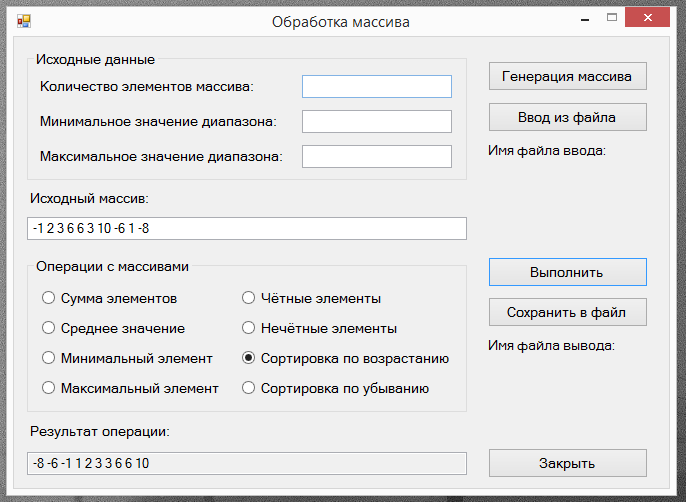


Рисунок 33 – Интерфейс программы

В таблице 1 представлены результаты тестирования функции программы, которая генерирует массив заданного размера в заданном диапазоне.

Таблица 1 – Результаты тестирования функции генерирования массива

|  |  |
| --- | --- |
| **Введенные значения** | **Результат** |
| 1) Количество элементов: 10  Минимальное значение диапазона: -10  Максимальное значение диапазона:10 | Исходный массив:  -1 2 3 6 6 3 10 -6 1 -8 |
| 2) Количество элементов: 0  Минимальное значение диапазона: 10  Максимальное значение диапазона: 11 | Вывод сообщения об ошибке:  «Введите корректное число элементов массива!» |
| 3) Количество элементов:  Минимальное значение диапазона:  Максимальное значение диапазона: | Вывод сообщения об ошибке:  «Введите числа!» |
| 4) Количество элементов: 1  Минимальное значение диапазона: -  Максимальное значение диапазона: - | Вывод сообщения об ошибке:  «Введите числа!» |
| 5) Количество элементов: 10  Минимальное значение диапазона: 10  Максимальное значение диапазона:0 | Вывод сообщения об ошибке:  «Минимальное значение массива больше максимального значения!» |
| 6) Количество элементов: 10  Минимальное значение диапазона: 5  Максимальное значение диапазона: 5 | Вывод сообщения об ошибке:  «Минимальное и максимальное значение равны» |
| 7) Количество элементов: 10  Минимальное значение диапазона: 0  Максимальное значение диапазона: 1 | Исходный массив:  0 1 1 0 0 0 1 0 1 0 |

На рисунке 34 представлен скриншот входного файла, из которого функция будет считываться исходный массив целых чисел. Для этого следует нажать кнопку «Ввод из файла», и выбрать файл через появляющееся диалоговое окно. В ходе выполнения функции каждая строка считывается в массив строк, после чего происходит обработка каждой отдельной строки от недопустимых символов и их последовательностей. Результат обработки представлен на рисунке 35.

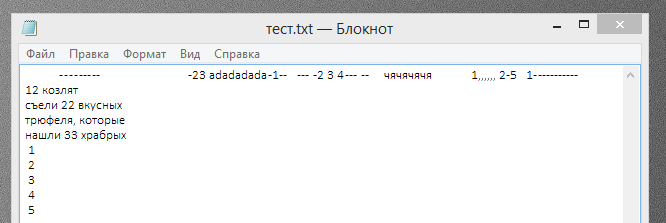


Рисунок 34 – Входной файл

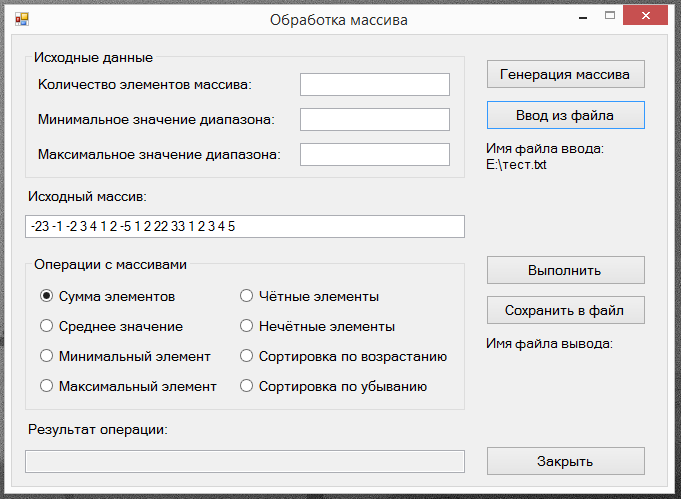


Рисунок 35 – Входной файл

На рисунке 36 представлен результат операции с исходным массивом. В качестве операции выбрана сортировка по возрастанию.

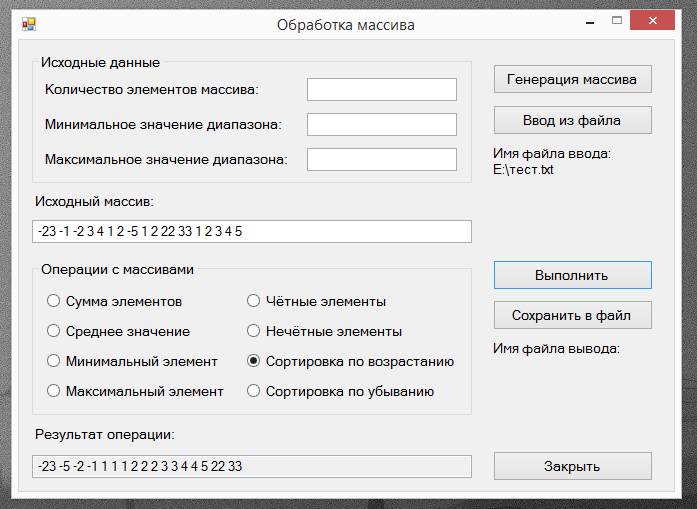


Рисунок 36 – Результат работы программы

В таблице 2 приведены результаты тестирования программы, при выборе операции, совершаемой с исходным массивом чисел.

Таблица 2 – Результаты тестирования выполнения выбранной операции над массивам

|  |  |  |
| --- | --- | --- |
| Исходный массив | Производимая операция | Результат |
| 23 -1 -2 3 4 1 2 -5 1 2 22 33 1 2 3 4 5 | Сумма элементов | 52 |
| Среднее значение | 3,05882352941176 |
| Минимальный элемент | -23 |
| Максимальный элемент | 33 |
| Чётные элементы | -2 4 2 2 22 2 4 |
| Нечётные элементы | -23 -1 3 1 -5 1 33 1 3 5 |
| Сортировка по возрастанию | -23 -5 -2 -1 1 1 1 2 2 2 3 3 4 4 5 22 33 |
| Сортировка по убыванию | 33 22 5 4 4 3 3 2 2 2 1 1 1 -1 -2 -5 -23 |
| 7 -7 4 5 -2 -5 -3 4 0 -6 | Сумма элементов | -3 |
| Среднее значение | -0,3 |
| Минимальный элемент | -7 |
| Максимальный элемент | 7 |
| Чётные элементы | 4 -2 4 0 -6 |
| Нечётные элементы | 7 -7 5 -5 -3 |
| Сортировка по возрастанию | -7 -6 -5 -3 -2 0 4 4 5 7 |
| Сортировка по убыванию | 7 5 4 4 0 -2 -3 -5 -6 -7 |

На рисунке 37 представлен результат поочерёдного сохранения всех операций программы.

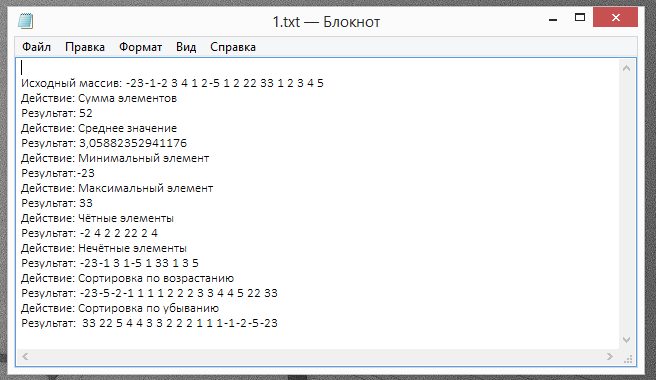


Рисунок 37 – Результат сохранения данных

На рисунке 38 представлен результат сохранения производимых операций с разными массивами.

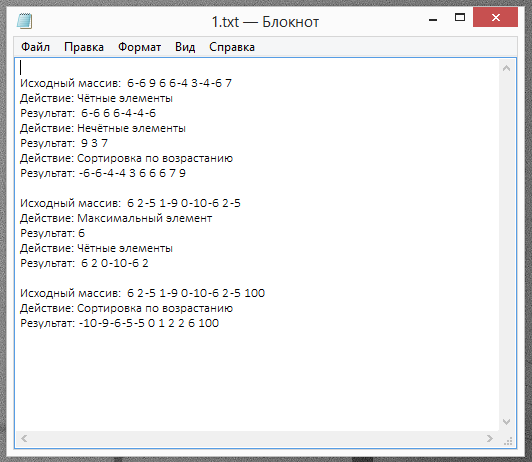


Рисунок 37 – Результат сохранения данных операций, производимых с разными массивами

## **Вывод**

В ходе данной лабораторной работы были получены практические знания, связанные с работой массивов и операциями, производимыми при работе с массивами. Помимо этих знаний также потребовались и знания по циклическим программам для реализации программы, выполняющей операции с массивами, которые были поставлены в условиях задачи. При этом были рассмотрены различные варианты возможной входной информации, предусмотрены некорректные значения вводимых данных.