Лабораторная работа №5

**«Базовые технологии обработки одномерных массивов»**

**1) Индивидуальное задание на разработку проекта**

Разработать программный код решения следующей задачи:

Дан динамический вещественный массив положительных и отрицательных чисел. Разработайте методы сортирующие исходный массив 2 алгоритмами: простого выбора и простого обмена.

**2) Техническое задание**

1. Добавить в существующую DLL-библиотеку функции, решающие предписанные задачи. Обмен данными между подпрограммами должен осуществляться через параметры, без использования глобальных переменных. Событийная процедура должна содержать только операторы вызова пользовательских подпрограмм.

2. В ходе решения задачи своего варианта предусмотреть написание функций для генерации массива, сортировки массива, создания нового массива, вывода массива. Для решения задачи своего варианта использовать многочисленные метод. материалы, которые перечислены в пояснениях (ЭОИС).

3. Подготовить тесты для контрольного решения задачи. Показать их использование, отобразив в отчете по заданию.

4. Выполнить проект и получить результаты, которые отобразить в виде рисунков при написании отчета по заданию.

5. Выполнить отчет в соответствии с ГОСТ на схемы алгоритмов и согласно ГОСТ оформления отчетов по НИР

**3) Схемы алгоритмов**

Реализую в DLL-библиотеки схемы алгоритмов, представленные в ЭИОС. Мной необходимо было реализовать алгоритмы под номерами 11 и 13. Представлю схемы алгоритмов на рисунках 1 и 2.

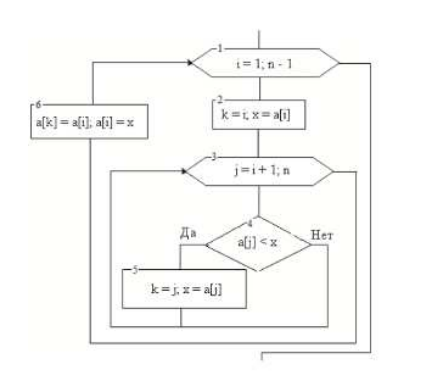
****

Рисунок 1 – Функция для сортировки массива простым выбором

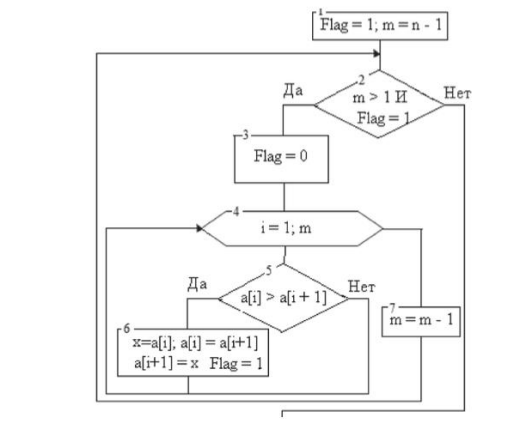
****

Рисунок 2 – Функция для сортировки массива простым обменом(2 способ)

**4) Программный код DLL-библиотеки**

В ходе создания DLL-библиотеки необходимо было написать код в двух файлах: заголовочном (.h) и исходном (.cpp).

Программный код заголовочного файла представлен ниже.

#pragma once

using namespace System;

using namespace System::Windows::Forms;

namespace dll\_kuleshov\_techprog\_sem3\_bei2202 {

public ref class Functions\_Class

{

public:

// ЛР #1

static double input(TextBox^ t);

static void output(double x, TextBox^ t);

static double get\_Radius(double L);

static double get\_Area(double L);

static void get\_Area\_mult(double L1, double& S1, double L2, double& S2, double L3, double& S3);

// ЛР #2

static void gen\_mas(double\* mas, int n);

static void output\_mas(double\* mas, int len, DataGridView^ grid);

static int count\_in\_bounds(double\* mas, int n);

static int set\_mas(double\* mas, int n, double\* rezmas);

// ЛР #3

literal System::String^ ConnectionString = "Provider=Microsoft.ACE.OLEDB.16.0;Data Source=1dim\_array.accdb";

static void CreateAccessDbTables();

static void CreateAccessDbFile();

static void FillAccessDb(double\* sourceArray, double\* transformedArray, int sourceArrayLength, int transformedArrayLength);

// ЛР #4

static void FillWordDocument(const double\* sourceArray, const double\* transformedArray, const int sourceArrayLength, const int transformedArrayLength);

// ЛР #5

static void ProstViborSort(double\* a, int); /// 11 algo

static void ProstObmenSort(double\* a, int n); /// 13 algo

};

}

Программный код исходного файла со всеми функциями представлен ниже.

#include "pch.h"

#include "dll\_kuleshov\_techprog\_sem3\_bei2202.h"

#include "Math.h"

#include <cstdlib>

#include <time.h>

using namespace System::Windows::Forms;

using namespace System;

using namespace System::IO;

using namespace System::Data::OleDb;

using namespace System::Runtime::InteropServices;

using namespace Microsoft::Office::Interop;

using namespace System::Globalization;

namespace dll\_kuleshov\_techprog\_sem3\_bei2202 {

double Functions\_Class::input(TextBox^ t) {

return Convert::ToDouble(t->Text);

}

void Functions\_Class::output(double x, TextBox^ t) {

t->Text = Convert::ToString(x);

}

double Functions\_Class::get\_Radius(double L) {

return L / (2 \* Math::PI);

}

double Functions\_Class::get\_Area(double L) {

double S = Math::PI \* Math::Pow(get\_Radius(L), 2);

return S;

}

void Functions\_Class::get\_Area\_mult(double L1, double& S1, double L2, double& S2, double L3, double& S3) {

S1 = Functions\_Class::get\_Area(L1);

S2 = Functions\_Class::get\_Area(L2);

S3 = Functions\_Class::get\_Area(L3);

}

// ЛР №2

void Functions\_Class::gen\_mas(double\* mas, int n) {

srand(time(0));

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

mas[i] = ((double)((int)(rand() % 600) - 300)) / 10.0;

}

}

int Functions\_Class::count\_in\_bounds(double\* mas, int n) {

int count = 0;

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

if (mas[i] >= 2 && mas[i] <= 10) {

count++;

}

}

return count;

}

void Functions\_Class::output\_mas(double\* mas, int len, DataGridView^ grid) {

grid->RowCount = 2;

grid->ColumnCount = len;

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

{

grid->Rows[0]->Cells[i]->Value = String::Format("[{0}]", i);

grid->Rows[1]->Cells[i]->Value = mas[i].ToString("F3");

}

}

int Functions\_Class::set\_mas(double\* mas, int n, double\* rezmas) {

int cou = count\_in\_bounds(mas, n);

int j = 0;

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

if (mas[i] < cou) {

rezmas[j] = mas[i];

j++;

}

}

return j;

}

// ЛР #3

void Functions\_Class::CreateAccessDbFile()

{

if (File::Exists(L"1dim\_array.accdb"))

File::Delete(L"1dim\_array.accdb");

ADOX::CatalogClass^ catalog = gcnew ADOX::CatalogClass();

catalog->Create(ConnectionString);

Marshal::FinalReleaseComObject(catalog->ActiveConnection);

Marshal::FinalReleaseComObject(catalog);

}

void Functions\_Class::CreateAccessDbTables()

{

OleDbConnection connection(ConnectionString);

connection.Open();

OleDbCommand command(LR"(

CREATE TABLE [ИсходныйМассив] (

[Id] LONG NOT NULL PRIMARY KEY,

[Value] DOUBLE NOT NULL);)", % connection);

command.ExecuteNonQuery();

command.CommandText = LR"(

CREATE TABLE [РезультирующийМассив] (

[Id] LONG NOT NULL PRIMARY KEY,

[Value] DOUBLE NOT NULL);)";

command.ExecuteNonQuery();

}

void Functions\_Class::FillAccessDb(double\* sourceArray, double\* transformedArray, int sourceArrayLength, int transformedArrayLength)

{

if (!File::Exists(L"1dim\_array.accdb"))

return;

OleDbConnection connection(ConnectionString);

connection.Open();

OleDbCommand command(String::Empty, % connection);

command.CommandText = L"DELETE FROM [ИсходныйМассив]";

command.ExecuteNonQuery();

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

{

command.CommandText = String::Format(LR"(

INSERT INTO [ИсходныйМассив]

([Id], [Value]) VALUES

({0}, {1});)", i, sourceArray[i].ToString(CultureInfo::InvariantCulture));

command.ExecuteNonQuery();

}

command.CommandText = L"DELETE FROM [РезультирующийМассив]";

command.ExecuteNonQuery();

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

{

command.CommandText = String::Format(LR"(

INSERT INTO [РезультирующийМассив]

([Id], [Value]) VALUES

({0}, {1});)", i, transformedArray[i].ToString(CultureInfo::InvariantCulture));

command.ExecuteNonQuery();

}

}

void Functions\_Class::FillWordDocument(const double\* sourceArray, const double\* transformedArray, const int sourceArrayLength, const int transformedArrayLength)

{

Object^ typeMissing = Type::Missing;

Object^ wdWord9TableBehavior = Word::WdDefaultTableBehavior::wdWord9TableBehavior;

Object^ wdAutoFitContent = Word::WdAutoFitBehavior::wdAutoFitContent;

Object^ wdGoToLine = Word::WdGoToItem::wdGoToLine;

Object^ wdGoToLast = Word::WdGoToDirection::wdGoToLast;

auto wordApplication = gcnew Word::ApplicationClass();

auto wordDocument = wordApplication->Documents->Add(typeMissing, typeMissing, typeMissing, typeMissing);

wordDocument->PageSetup->Orientation = Word::WdOrientation::wdOrientLandscape;

wordDocument->PageSetup->TopMargin = wordApplication->CentimetersToPoints(1.25);

wordDocument->PageSetup->RightMargin = wordApplication->CentimetersToPoints(1.25);

wordDocument->PageSetup->BottomMargin = wordApplication->CentimetersToPoints(1.25);

wordDocument->PageSetup->LeftMargin = wordApplication->CentimetersToPoints(1.25);

wordApplication->Selection->TypeText(L"Исходный массив:");

wordApplication->Selection->TypeParagraph();

auto wordTable = wordDocument->Tables->Add(wordApplication->Selection->Range, 2, sourceArrayLength, wdWord9TableBehavior, wdAutoFitContent);

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

{

wordTable->Cell(1, i + 1)->Range->InsertAfter(String::Format(L"[{0}]", i));

wordTable->Cell(2, i + 1)->Range->InsertAfter(sourceArray[i].ToString(L"F3"));

}

wordApplication->Selection->GoTo(wdGoToLine, wdGoToLast, typeMissing, typeMissing);

wordApplication->Selection->TypeParagraph();

wordApplication->Selection->TypeText(L"Результирующий массив:");

wordApplication->Selection->TypeParagraph();

wordTable = wordDocument->Tables->Add(wordApplication->Selection->Range, 2, transformedArrayLength, wdWord9TableBehavior, wdAutoFitContent);

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

{

wordTable->Cell(1, i + 1)->Range->InsertAfter(String::Format(L"[{0}]", i));

wordTable->Cell(2, i + 1)->Range->InsertAfter(transformedArray[i].ToString(L"F3"));

}

wordDocument->Content->Font->Name = L"Times New Roman";

wordDocument->Content->Font->Size = 12;

wordApplication->Selection->GoTo(wdGoToLine, wdGoToLast, typeMissing, typeMissing);

try {

Object^ filename = Path::Combine(Directory::GetCurrentDirectory(), L"1dim\_array.docx");

wordDocument->SaveAs(filename, typeMissing,

typeMissing, typeMissing, typeMissing,

typeMissing, typeMissing, typeMissing,

typeMissing, typeMissing, typeMissing,

typeMissing, typeMissing, typeMissing,

typeMissing, typeMissing);

wordApplication->Visible = true;

}

catch(Exception^ ex){

}

}

void Functions\_Class::ProstViborSort(double\* a, int n) {

for (int i = 0; i < n - 1; ++i) {

int k = i;

double x = a[i];

for (int j = i + 1; j < n; ++j) {

if (a[j] < x) {

k = j;

x = a[j];

}

}

a[k] = a[i];

a[i] = x;

}

}

void Functions\_Class::ProstObmenSort(double\* a, int n) {

bool Flag = 1;

int m = n - 1;

while (m > 0 && Flag == 1) {

Flag = 0;

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

if (a[i] > a[i + 1]) {

double x = a[i];

a[i] = a[i + 1];

a[i + 1] = x;

Flag = 1;

}

}

--m;

}

}

}

**5) Программный код событийной процедуры**

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ g = Microsoft::VisualBasic::Interaction::InputBox("Введите количество элементов в массиве",

"Ввод", "", -1, -1);

int n;

double\* mas;

double\* rezmas;

double\* to\_sort1;

double\* to\_sort2;

try {

n = Convert::ToInt16(g);

mas = new double[n];

rezmas = new double[n];

to\_sort1 = new double[n];

to\_sort2 = new double[n];

}

catch (System::FormatException^ e) {

MessageBox::Show("Ошибка при вводе данных", "Сообщение", MessageBoxButtons::OK, MessageBoxIcon::Error);

return;

}

Functions\_Class::gen\_mas(mas, n);

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

to\_sort1[i] = to\_sort2[i] = mas[i];

}

Functions\_Class::output\_mas(mas, n, dataGridView1);

int sz = Functions\_Class::set\_mas(mas, n, rezmas);

Functions\_Class::output\_mas(rezmas, sz, dataGridView2);

Functions\_Class::FillWordDocument(mas, rezmas, n, sz);

Functions\_Class::FillAccessDb(mas, rezmas, n, sz);

Functions\_Class::FillWordDocument(mas, rezmas, n, sz);

label3->Text = "Количество чисел в диапазоне[2; 10]: " + Functions\_Class::count\_in\_bounds(mas, n);

Functions\_Class::ProstViborSort(to\_sort1, n);

Functions\_Class::ProstObmenSort(to\_sort2, n);

Functions\_Class::output\_mas(to\_sort1, n, dataGridView3);

Functions\_Class::output\_mas(to\_sort2, n, dataGridView4);

delete[] mas;

delete[] rezmas;

delete[] to\_sort1;

delete[] to\_sort2;

}

**6) Результаты работы программы**

На рисунке 3 представлен результат работы программы при выдуманном размере массива. Как видно, программа работает корректно.

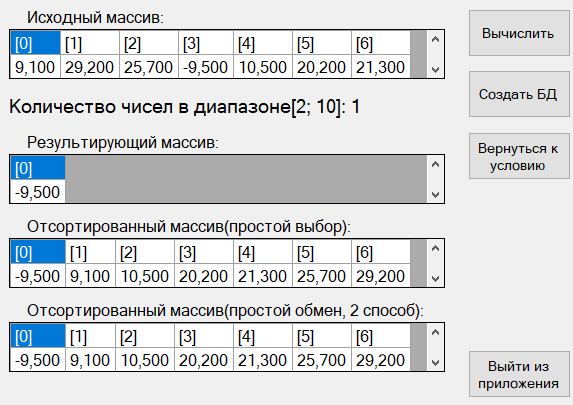


Рисунок 3 – Результаты работы при тестовых данных

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Гуриков С. Р. Основы алгоритмизации и программирования на Visual C++ : учебное пособие / С.Р. Гуриков. — Москва: ИНФРА-М, 2022. — 515 с. — (Высшее образование: Бакалавриат). — DOI 10.12737/1039154. – ISBN 978-5-16-015500-5. – Текст: электронный. – URL: https://znanium.com/catalog/product/1039154 – Режим доступа: по подписке.