

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА - Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта

Кафедра математического обеспечения и стандартизации

информационных технологий

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1**

**Реализация структуры данных задачи на одномерном массиве по дисциплине**

«СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ»

|  |  |
| --- | --- |
| Выполнил студень группы ИНБО-02-21      Принял старший преподаватель | Юдов С.А. |
| кафедры МОСИТ | Скворцова Л.А. |
| Практическая работа выполнена «\_\_»\_\_\_\_\_\_\_2022г. | \_\_\_\_\_\_\_\_\_ |
|  |  |
| «Зачтено» «\_\_»\_\_\_\_\_\_\_2022г. | \_\_\_\_\_\_\_\_\_ |
|  |  |

Москва 2022

**Содержание**

[Задание 1 3](#_Toc96870102)

[1. Условие задачи и варианта 3](#_Toc96870103)

[2. Разработка задачи 3](#_Toc96870104)

[3. Декомпозиция 6](#_Toc96870105)

[4. Определение функций 6](#_Toc96870106)

[5. Реализация функций 7](#_Toc96870107)

[6. Кодирование алгоритма программы 9](#_Toc96870108)

[7. Таблица тестов программы 10](#_Toc96870109)

[Задание 2 10](#_Toc96870110)

[1. Условие задачи и варианта 10](#_Toc96870111)

[2. Декомпозиция 11](#_Toc96870112)

[3. Определение функций 11](#_Toc96870113)

[4. Реализация функций 12](#_Toc96870114)

[5. Кодирование алгоритма программы 14](#_Toc96870115)

[6. Таблица тестов программы 15](#_Toc96870116)

[Задание 3 15](#_Toc96870117)

[1. Условие задачи и варианта 15](#_Toc96870118)

[2. Декомпозиция 15](#_Toc96870119)

[3. Определение функций 16](#_Toc96870120)

[4. Реализация функций 17](#_Toc96870121)

[5. Кодирование алгоритма программы 18](#_Toc96870122)

[6. Таблица тестов программы 19](#_Toc96870123)

[Вывод 20](#_Toc96870124)

[Список информационных источников 21](#_Toc96870125)

# Задание 1

## 1. Условие задачи и варианта

1.1. Дан статический массив из целых элементов.

* + 1. Определить сколько в массиве простых чисел Мерсенна. Считать натуральное число M простым числом Мерсенна, если оно удовлетворяет свойствам: 1) М – простое число 2) число М+1 является степенью двойки. Например, число М=31
    2. Удалить минимальное число, которое является степенью степень числа 2
    3. Вставить в массив новый элемент после элемента значение которого является максимальным простым числом Мерсенна.

## 2. Разработка задачи

2.1. Постановка задачи.

2.1.1. Дано. Дан массив из n элементов целого типа int.

2.1.2. Результат. Определить сколько в массиве простых чисел Мерсенна по заданным правилам.

2.1.3. Ограничения. Массив натуральных чисел.

2.2. Описание модели решения

Исходный массив a статически максимального размера N = 1000.

Текущий размер n массива mas определяет пользователь n <= N.

Число Мерсенна – это числа вида , где n – натуральное число

2.3. Декомпозиция – список алгоритмов, которые требуются разработать в соответствии исследованной моделью

2.3.1. Список подзадач:

1. Проверить, является ли число x простым и x + 1 степенью двойки
2. Подсчёт количества таких чисел в массиве
3. Поиск и удаление минимального элемента степени двойки
4. Поиск максимального числа Мерсенна и вставка любого числа перед ним
5. Вывод массива в консоль.
6. Заполнение массива с указанием его длины с клавиатуры.

2.3.2. Определение прототипов функций:

1. Заполнение исходного массива значениями с клавиатуры

Предусловие. n – число заполняемых элементов, 0<=n<=1000 Постусловие. Заполненный массив из n элементов void inputArray(int\* x, int n)

1. Вывод значений массива

### Предусловие. n>0

Постусловие. Вывод значений масива void outArray(int\* x, int x)

1. Функции декомпозиции

***//*** Определение является ли число степенью двойки

Предусловие. x > 0

Постусловие. Результат логическое значение.

bool degreeoftwo(int entered)

***//*** Определение является ли число простым

### Предусловие. x > 0

Постусловие. Результат логическое значение

bool prost(int n)

***//*** Определение является ли число числом Мерсенна

Предусловие. x > 0

Постусловие. Результат логическое значение.

bool checkMersen(int x)

***//Проход по массиву,*** определение количества чисел Мерсенна в массиве

Предусловие. n >0

Постусловие. Массив из size элементов int checkMersenN(int \*x, int n)

// Добавление нового элемента в любое место массива

Предусловие. size>0, currid > -1

Постусловие. Массив из size элементов, значение новой переменной value, индекс вставки currid

void addelem(int \*&arr, int &size, const int value, int currid)

// Удаление любого элемента из списка

Предусловие. size >0

Постусловие. Массив из size элементов, индекс удаляемого элемента id

void delelem(int \*&arr, int &size, int id)

// Поиск и удаление минимального элемента степени двойки

Предусловие. n>0

Постусловие. Массив из size элементов, размер массива N

void findmindel(int \*&x, int &n)

// Поиск максимального числа Мерсенна и вставка любого числа перед ним

Предусловие. n>0

Постусловие. Массив из size элементов, размер массива N

void findmaxins(int \*&x, int &n, int nw)

2.4. Разработка алгоритмов операций и представление его на псевдокоде

1. Алгоритмы ввода и вывода массива определим при реализации
2. Алгоритмы задач декомпозиции

Алгоритм функции void findmindel(int \*x, int &n)

void findmindel(int \*x, int &n) {

bool flag = false;

int min\_i = -1

long long int min = 1000000000000;

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

if (x[i] < min && degreeoftwo(x[i])) {

min\_i = i;

min = x[i];

}

}

if (min\_i > -1) {

for (int i = min\_i; i < n; i++) {

x[i] = x[i + 1];

}

n--;

}

}

Алгоритм функции void findmaxins(int \*x, int &n, int nw)

void findmaxins(int \*x, int &n, int nw) {

int max\_i = -1;

int max = 0;

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

if (x[i] > max && checkMersen(x[i])) {

max\_i = i;

max = x[i];

}

}

if (max > -1) {

for (int i = n; i > max\_i; i--) {

x[i + 1] = x[i];

}

x[max\_i + 1] = nw;

}

n++;

}

## 3. Декомпозиция

Задачу следует разбить на следующие подзадачи:

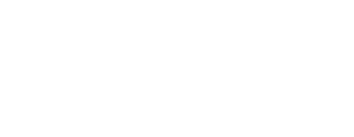
1. Проверить, является ли число x простым и x + 1 степенью двойки
2. Подсчёт количества таких чисел в массиве
3. Поиск и удаление минимального элемента степени двойки
4. Поиск максимального числа Мерсенна и вставка любого числа перед ним
5. Вывод массива в консоль.
6. Заполнение вектора с указанием его длины с клавиатуры.

## 4. Определение функций

4.1. Вывод массива в консоль производится с помощью процедуры:



A



x

-

int\*

n

-

int



Входные данные



Результат

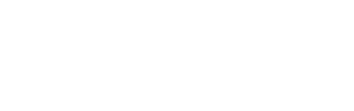


void outarray

4.2. Заполнение массива с указанием его длины с клавиатуры производится с помощью процедуры:



A



x

-

int\*

n

-

int



Входные данные



Результат

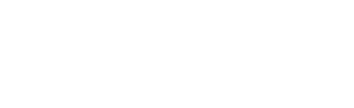


void inputarray

4.3. Определение количества чисел Мерсенна в массиве:



A



x

-

int



Входные данные



Результат



int

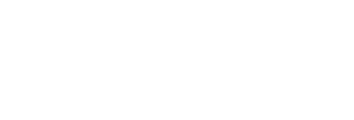
checkMersenN

Логическое значение

4.4. Поиск и удаление минимального элемента степени двойки:



A



x

-

int\*

n

-

int



Входные

данные



Результат

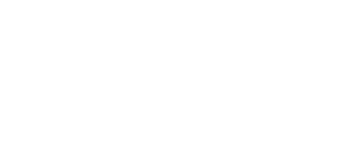


void findmindel

4.5. Поиск максимального числа Мерсенна и вставка любого числа перед ним



A



x

-

int\*

nw

-

int

n

-

int



Входные данные



Результат



void

findmaxins

4.6. Добавление нового элемента в любое место массива:

-

id

int delelem \*

int delelem \*

size

-

nw

-

int

int

-

value

size

int

-

4.7.

Удаление любого элемента из списка

:



A



arr

-

int\*



Входные данные



Результат



void addelem



A

x

-

int\* delelem \*



Входные данные



Результат



void delelem

## 5. Реализация функций

bool degreeoftwo(int entered) {

int temp = 1;

while (temp < entered) {

temp \*= 2;

}

if (temp == entered) {

return true;

} else {

return false;

}

}

bool prost(int n) {

for (int i = 2; i < n; i++) {

if (n % i == 0) {

return false;

}

return true;

}

}

bool checkMersen(int x) {

if (prost(x) && degreeoftwo(x + 1)) {

return true;

}

return false;

}

int checkMersenN(int \*x, int n) {

int m = 0;

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

if (checkMersen(x[i])) {

m += 1;

}

}

return m;

}

void findmindel(int \*x, int &n) {

bool flag = false;

int min\_i = -1

long long int min = 1000000000000;

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

if (x[i] < min && degreeoftwo(x[i])) {

min\_i = i;

min = x[i];

}

}

if (min\_i > -1) {

for (int i = min\_i; i < n; i++) {

x[i] = x[i + 1];

}

n--;

}

}

void findmaxins(int \*x, int &n, int nw) {

int max\_i = -1;

int max = 0;

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

if (x[i] > max && checkMersen(x[i])) {

max\_i = i;

max = x[i];

}

}

if (max > -1) {

for (int i = n; i > max\_i; i--) {

x[i + 1] = x[i];

}

x[max\_i + 1] = nw;

}

n++;

}

void inputarray(int \*x, int n) {

cout << "Wedite " << n << " chisel\n";

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

cin >> x[i];

}

}

void outarray(int \*x, int n) {

cout << "Massiv " << n << " chisel\n";

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

cout << x[i] << ' ';

}

cout << endl;

}

void inputRandarray(int \*x, int n) {

srand(time(0));

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

x[i] = rand() % 100;

}

}

## 6. Кодирование алгоритма программы

#include "iostream"

using namespace std;

const int N = 1000;

int main(){

int a[N];

inputarray(a, size);

cout << "Zadanie 1.1 " << checkMersenN(a, size) << endl;

cout << "Zadanie 1.2 " << endl;

findmindel(a, size);

outarray(a, size);

cout << "Zadanie 1.3 " << endl;

findmaxins(a, size, 5555);

}

## 7. Таблица тестов программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер задачи | Исходные данные | Ожидаемый результат | Результат программы | Тест пройден/не пройден |
| 1 | size = 8  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  checkMersenN(a, size) | 2 | 2 | Тест пройден |
| 2 | size = 8  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  findmindel(a, n); | 0 3 5 7 10 12 16 | 0 3 5 7 10 12 16 | Тест пройден |
| 3 | size = 8  nw = 5555  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  findmaxins(a, size, nw); | 0 3 5 7 5555 10 12 16 | 0 3 5 7 5555 10 12 16 | Тест пройден |
| Неверный ввод | N = 0 | Вывод сообщения об ошибке | Неверный ввод, введите n в диапазоне от 1 до 1000 | Пройден |

# Задание 2

## 1. Условие задачи и варианта

1.1. Дан динамический массив из целых элементов.

* + 1. Определить сколько в массиве простых чисел Мерсенна. Считать натуральное число M простым числом Мерсенна, если оно удовлетворяет свойствам: 1) М – простое число 2) число М+1 является степенью двойки. Например, число М=31
    2. Удалить минимальное число, которое является степенью степень числа 2
    3. Вставить в массив новый элемент после элемента значение которого является максимальным простым числом Мерсенна.

возрастающую последовательность.

## 2. Декомпозиция

Задачу следует разбить на следующие подзадачи:

2.1 Проверить, является ли число x простым и x + 1 степенью двойки

2.2 Подсчёт количества таких чисел в массиве

2.3 Поиск и удаление минимального элемента степени двойки

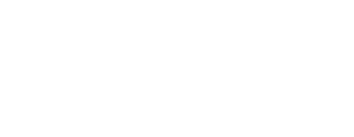
2.4 Поиск максимального числа Мерсенна и вставка любого числа перед ним

## 3. Определение функций

3.1. Вывод массива в консоль производится с помощью процедуры:



A



x

-

int\*

n

-

int



Входные данные



Результат

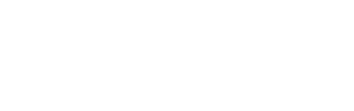


void outarray

3.2. Заполнение массива с указанием его длины с клавиатуры производится с помощью процедуры:



A



x

-

int\*

n

-

int



Входные данные



Результат

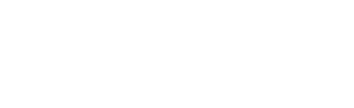


void inputarray

3.3. Определение количества чисел Мерсенна в массиве:



A



x

-

int



Входные данные



Результат



int

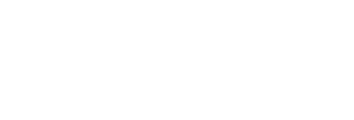
checkMersenN

Логическое значение

3.4. Поиск и удаление минимального элемента степени двойки:



A



x

-

int\*

n

-

int



Входные

данные



Результат

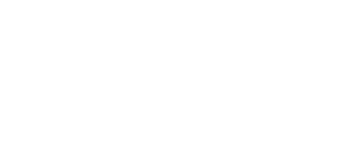


void findmindel

3.5. Поиск максимального числа Мерсенна и вставка любого числа перед ним



A



x

-

int\*

nw

-

int

n

-

int



Входные данные



Результат



void

findmaxins

3.6. Добавление нового элемента в любое место массива:

-

id

int delelem \*

int delelem \*

size

-

nw

-

int

int

-

value

size

int

-

3.7.

Удаление любого элемента из списка

:



A



arr

-

int\*



Входные данные



Результат



void addelem



A

x

-

int\* delelem \*



Входные данные



Результат



void delelem

## 4. Реализация функций

bool degreeoftwo(int entered) {

int temp = 1;

while (temp < entered) {

temp \*= 2;

}

if (temp == entered) {

return true;

} else {

return false;

}

}

bool prost(int n) {

for (int i = 2; i < n; i++) {

if (n % i == 0) {

return false;

}

return true;

}

}

bool checkMersen(int x) {

if (prost(x) && degreeoftwo(x + 1)) {

return true;

}

return false;

}

int checkMersenN(int \*x, int n) {

int m = 0;

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

if (checkMersen(x[i])) {

m += 1;

}

}

return m;

}

void addelem(int \*&arr, int &size, const int value, int currid) {

size++;

arr = (int \*) realloc(arr, sizeof(int) \* (size));

if (!arr) exit(1);

for (int i = size - 1; i > currid; i--) {

arr[i] = arr[i - 1];

}

arr[currid] = value;

}

void delelem(int \*&arr, int &size, int id) {

for (int i = id; i < size; i++) {

arr[i] = arr[i + 1];

}

size--;

arr = (int \*) realloc(arr, sizeof(int) \* size);

}

void findmindel(int \*&x, int &n) {

bool flag = false;

min\_i = -1;

long long int min = 1000000000000;

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

if (x[i] < min && degreeoftwo(x[i])) {

min\_i = i;

min = x[i];

flag = true;

}

}

if (min\_i > -1) {

delelem(x,n,min);

}

}

void findmaxins(int \*&x, int &n, int nw) {

int max\_i = -1;

int max = 0;

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

if (x[i] > max && checkMersen(x[i])) {

max\_i = i;

max = x[i]

}

}

if (max\_i > -1) {

addelem(x,n,nw,max + 1);

}

}

void inputarray(int \*x, int n) {

cout << "Wedite " << n << " chisel\n";

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

cin >> x[i];

}

}

void outarray(int \*x, int n) {

cout << "Massiv " << n << " chisel\n";

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

cout << x[i] << ' ';

}

cout << endl;

}

## 5. Кодирование алгоритма программы

#include <iostream>

#include <cstdlib>

using namespace std;

int main() {

int size = 8;

int \*dinarr = (int \*) malloc(sizeof(int) \* size);

inputarray(dinarr, size);

outarray(dinarr, size);

cout << "Zadanie 1.1 " << checkMersenN(dinarr, size) << endl;

cout << "Zadanie 1.2 " << endl;

findmindel(dinarr, size);

outarray(dinarr, size);

cout << "Zadanie 1.3 " << endl;

findmaxins(dinarr, size, 5555);

outarray(dinarr,size);

free(dinarr);

dinarr = nullptr;

return 0;

}

## 6. Таблица тестов программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | Ожидаемый результат | Результат программы | Тест пройден/не пройден |
| 1 | size = 8  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  checkMersenN(a, size) | 2 | 2 | Тест пройден |
| 2 | size = 8  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  findmindel(a, n); | 0 3 5 7 10 12 16 | 0 3 5 7 10 12 16 | Тест пройден |
| 3 | size = 8  nw = 5555  a[8] = {0, 3, 5, 8, 7, 10, 12, 16}  findmaxins(a, size, nw); | 0 3 5 7 5555 10 12 16 | 0 3 5 7 5555 10 12 16 | Тест пройден |

# Задание 3

## 1. Условие задачи и варианта

1.1. Дан вектор из целых элементов.

* + 1. Определить сколько в массиве простых чисел Мерсенна. Считать натуральное число M простым числом Мерсенна, если оно удовлетворяет свойствам: 1) М – простое число 2) число М+1 является степенью двойки. Например, число М=31
    2. Удалить минимальное число, которое является степенью степень числа 2
    3. Вставить в массив новый элемент после элемента значение которого является максимальным простым числом Мерсенна.

## 2. Декомпозиция

Задачу следует разбить на следующие подзадачи:

2.1. Вывод вектора в консоль.

2.2. Заполнение вектора с указанием его длины с клавиатуры.

2.3 Проверить, является ли число x простым и x + 1 степенью двойки

2.4 Подсчёт количества таких чисел в массиве

2.5 Поиск и удаление минимального элемента степени двойки

2.6 Поиск максимального числа Мерсенна и вставка любого числа перед ним

2.7 Определение количества чисел исходного массива, цифры которого образуют возрастающую последовательность.

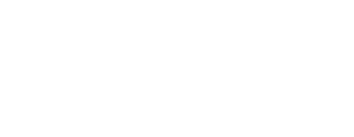
## 3. Определение функций

3.1. Вывод вектора в консоль производится с помощью процедуры:

&x vector<int>



A



Входные данные



Результат



void outputvector

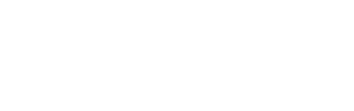
3.2. Заполнение вектора с указанием его длины с клавиатуры производится с помощью процедуры:

&x vector<int>

size int



A



Входные данные



Результат



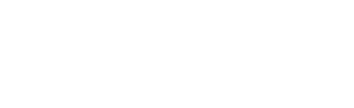
void inputvector

3.3. Определение количества чисел Мерсенна в массиве:

&x vector<int>



A



Входные данные



Результат



int

checkMersenN

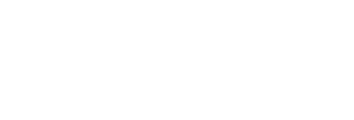
Логическое значение

3.4. Поиск и удаление минимального элемента степени двойки:

&x vector<int>



A



Входные

данные



Результат



void findmindel

3.5. Поиск максимального числа Мерсенна и вставка любого числа перед ним

&x vector<int>



A

nw

-

int



Входные данные



Результат



void

findmaxins

## 4. Реализация функций

bool degreeoftwo(int entered) {

int temp = 1;

while (temp < entered) {

temp \*= 2;

}

if (temp == entered) {

return true;

} else {

return false;

}

}

bool prost(int n) {

for (int i = 2; i < n; i++) {

if (n % i == 0) {

return false;

}

return true;

}

}

bool checkMersen(int x) {

if (prost(x) && degreeoftwo(x + 1)) {

return true;

}

return false;

}

int checkMersenN(vector<int> &x) {

int m = 0;

for (vector<int>::iterator i = x.begin(); i != x.end(); i++) {

if (checkMersen(\*i)) {

m += 1;

}

}

return m;

}

void findmindel(vector<int> &x) {

bool flag = false;

long long int min = 1000000000000;

int min\_i = -1;

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

if (x[i] < min && degreeoftwo(x[i])) {

min\_i = i;

min = x[i];

}

}

if (min\_i > -1) {

x.erase(x.begin() + min\_i);

}

}

void findmaxins(vector<int> &x, const int nw) {

int max\_i = -1;

int max = -1;

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

if (x[i] > max && checkMersen(x[i])) {

max\_i = i;

max = x[i];

}

}

if (max\_i > -1) {

x.insert(x.begin() + max\_i + 1, nw);

}

}

void inputvector(vector<int> &x, int size) {

cout << "Wedite " << size << " chisel\n";

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

int temp;

cin >> temp;

x[i] = temp;

}

}

void outputvector(vector<int> &x) {

cout << "Massiv " << x.size() << " chisel\n";

for (auto const &element: x) {

cout << element << ' ';

}

cout << endl;

}

## 5. Кодирование алгоритма программы

#include <iostream>

#include <vector>

using namespace std;

int main() {

int size = 8;

vector<int> myVector(size);

inputvector(myVector, size);

outputvector(myVector);

cout << "Zadanie 1.1 " << checkMersenN(myVector) << endl;

cout << "Zadanie 1.2 " << endl;

findmindel(myVector);

outputvector(myVector);

cout << "Zadanie 1.3 " << endl;

findmaxins(myVector, 5555);

outputvector(myVector);

return 0;

}

## 6. Таблица тестов программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | Ожидаемый результат | Результат программы | Тест пройден/не пройден |
| 1 | size = 8  vector<int> myVector = {0, 3, 5, 8, 7, 10, 12, 16}  checkMersenN(a) | 2 | 2 | Тест пройден |
| 2 | vector<int> myVector = {0, 3, 5, 8, 7, 10, 12, 16}  findmindel(a); | 0 3 5 7 10 12 16 | 0 3 5 7 10 12 16 | Тест пройден |
| 3 | size = 8  nw = 5555  vector<int> myVector = {0, 3, 5, 8, 7, 10, 12, 16}  findmaxins(a, nw); | 0 3 5 7 5555 10 12 16 | 0 3 5 7 5555 10 12 16 | Тест пройден |

# Вывод

В течение выполнения данной работы, мной были получены знания по работе со стандартными типами данных языка для представления многоэлементных однородных структур данных задачи в программе. Также были приобретены навыки создания алгоритмов операций над одномерными массивами и по их реализации.

# Список информационных источников

* Учебник по C++ [http://www.lmpt.univ-tours.fr/~volkov/C++.pdf](https://d.docs.live.net/793bf2b94bcfd004/Рабочий%20стол/http:/www.lmpt.univ-tours.fr/~volkov/C++.pdf)
* Документация по языку C++ [https://docs.microsoft.com/ruru/cpp/?view=msvc-160](https://d.docs.live.net/793bf2b94bcfd004/Рабочий%20стол/https:/docs.microsoft.com/ruru/cpp/?view=msvc-160)