

**МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ**

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Организация ЭВМ и систем»

**Тема: Организация связи Ассемблера с ЯВУ на примере
программы построения частотного распределение попаданий
псевдослучайных целых чисел в заданные интервалы.**

Вариант 18

Студент гр. 1303

Насонов Я.К.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Рассмотреть способ организации связи Ассемблера с ЯВУ. Разработать программу построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

Задание.

На языке С программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения.

Далее должны вызываться 1 или 2 ассемблерных процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должны вызываться как независимо скомпилированные модули. Передача параметров в процедуру должна выполняться через кадр стека.

Результирующий массив частотного распределения чисел по интервалам, сформированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле и выводится на экран средствами ЯВУ.

Выполнение работы.

Main.cpp

В функции main() происходит считывание данных и их подготовка к передаче в ассемблерный блок. А также обрабатываются следующие необходимые по условию исключения:

Количество интервалов должно быть от 0 до 24

Количество интервалов должно быть меньше макс. знач. – мин. знач.

Все левые границы должна быть больше мин. знач.

Правая граница последнего интервала должна быть больше макс. знач.

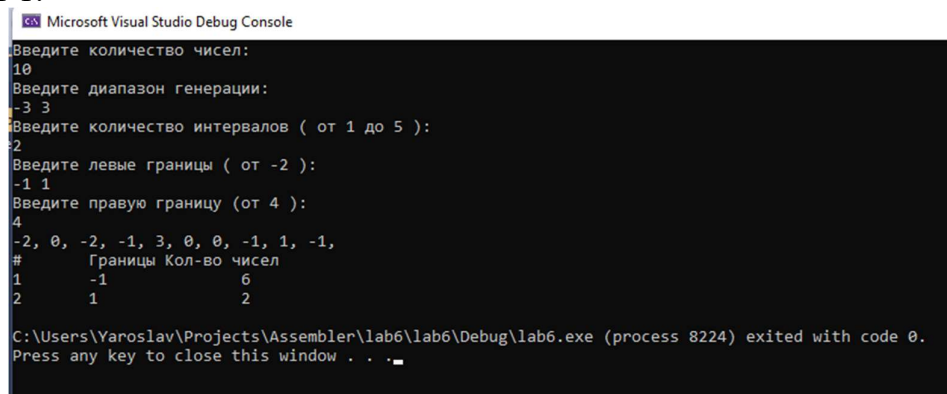
После вызова ассемблерного блока выводится результат его выполнения.

Module.asm

В блоке происходит поиск количества чисел, вошедших в каждый заданный интервал. Поиск реализован в виде двойного цикла. В первом происходит последовательный выбор из массива сгенерированных чисел. В во втором - перебор левых границ для нахождения нужного диапазона. Перед вторым циклом выбранное число проверяется на вхождение в диапазон между правой границей и макс. знач., в случае положительного исхода - число пропускается, так как оно не попадает ни в один из диапазонов. Иначе левые границы перебираются в порядке убывания. После выхода из внутреннего цикла из массива result для вывода достается число, соответствующее найденному диапазону, увеличивается на один и кладется обратно. После прохождения циклов полученный массив result будет использоваться для вывода.

Тестирование.

Тест 1.



```
Microsoft Visual Studio Debug Console
Введите количество чисел:
10
Введите диапазон генерации:
-3 3
Введите количество интервалов ( от 1 до 5 ):
2
Введите левые границы ( от -2 ):
-1 1
Введите правую границу (от 4 ):
4
-2, 0, -2, -1, 3, 0, 0, -1, 1, -1,
#      Границы Кол-во чисел
1      -1          6
2      1           2

C:\Users\Yaroslav\Projects\Assembler\lab6\lab6\Debug\lab6.exe (process 8224) exited with code 0.
Press any key to close this window . . .
```

Тест 2.

```
Microsoft Visual Studio Debug Console
Введите количество чисел:
5
Введите диапазон генерации:
-3 3
Введите количество интервалов ( от 1 до 5 ):
5
Количество интервалов должно быть ( от 1 до 5 ).
C:\Users\Yaroslav\Projects\Assembler\lab6\lab6\Debug\lab6.exe (process 32) exited with code 0.
Press any key to close this window . . .
```

Тест 3.

```
Microsoft Visual Studio Debug Console
Введите количество чисел:
10
Введите диапазон генерации:
-3 3
Введите количество интервалов ( от 1 до 5 ):
4
Введите левые границы ( от -2 ):
-3 -2 -1 0
Все границы должны быть > мин. знач.
C:\Users\Yaroslav\Projects\Assembler\lab6\lab6\Debug\lab6.exe (process 15684) exited with code 0.
Press any key to close this window . . .
```

Вывод.

Рассмотрен способ организации связи Ассемблера с ЯВУ. Составлена программа построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <random>
```

```
#include <string>
```

```
using namespace std;
```

```
extern "C" void func(int* intervals, int interval_count, int number_count,  
int* numbers, int* result);
```

```
int main() {
```

```
    system("chcp 1251 > nul");
```

```
    setlocale(LC_CTYPE, "rus");
```

```
    int number_count, min_value, max_value, interval_count;
```

```
    cout << "Введите количество чисел:" << endl;
```

```
    cin >> number_count;
```

```
    cout << "Введите диапазон генерации:" << endl;
```

```
    cin >> min_value >> max_value;
```

```
    int max_intervals = max_value - min_value - 1;
```

```
    cout << "Введите количество интервалов ( от 1 до " << max_intervals  
<< " ):" << endl;
```

```
    cin >> interval_count;
```

```
    if (interval_count <= 0 || interval_count > abs(max_intervals)) {
```

```
        cout << "Количество интервалов должно быть ( от 1 до " <<  
max_intervals << " )." << endl;
```

```
        return 0;
```

```
    }
```

```
    cout << "Введите левые границы ( от " << min_value + 1 << " ):" <<  
endl;
```

```
    int* intervals = new int[interval_count + 1];
```

```
    for (int i = 0; i < interval_count; ++i) {
```

```
        cin >> intervals[i];
```

```
    }
```

```
    for (int i = 0; i < interval_count; i++) {
```

```
        for (int j = i; j < interval_count; j++) {
```

```
            if (intervals[i] > intervals[j]) {
```

```
                swap(intervals[i], intervals[j]);
```

```

        }
    }
}

if (intervals[0] <= min_value) {
    cout << "Все границы должны быть > мин. знач." << endl;
    return 0;
}

cout << "Введите правую границу (от " << max_value + 1 << "):" <<
endl;
cin >> intervals[interval_count];

if (intervals[interval_count] < intervals[interval_count - 1]) {
    cout << "Правая граница посл. интервала должна быть >=
левой границы посл. интервала." << endl;
    return 0;
}

if (intervals[interval_count] <= max_value) {
    cout << "Правая граница последнего интервала должна быть
> макс. знач." << endl;
    return 0;
}

int *numbers = new int[number_count];
random_device rd;
mt19937 generator(rd());

uniform_int_distribution<> dist(min_value, max_value);
for (int i = 0; i < number_count; i++) {
    numbers[i] = dist(generator);
    cout << numbers[i] << ", ";
}
cout << endl;

int* result = new int[interval_count];

for (int i = 0; i < number_count; i++) {
    result[i] = 0;
}

func(intervals, interval_count, number_count, numbers, result);

ofstream file("output.txt");
string info = "#\tГраницы\tКол-во чисел";

```

```

        file << info << endl;
        cout << info << endl;
        for (int i = 0; i < interval_count; i++) {
            string row = to_string(i + 1) + "\t" + to_string(intervals[i]) + "\t\t"
+ to_string(result[i]) + "\n";
            file << row;
            cout << row;
        }
        return 0;
    }
}

```

Название файла: module.asm

```

.MODEL FLAT, C
.CODE

```

```

PUBLIC C func
func PROC C intervals: dword, interval_count: dword, number_count:
dword, numbers: dword, result: dword
    push esi
    push edi
    push eax
    push ebx
    push ecx
    push edx

    mov esi, numbers
    mov edi, result
    mov eax, 0

    start:
        mov ebx, [esi + 4*eax] ; берем число из массива
сгенерированных чисел
        push esi
        mov ecx, interval_count

        mov esi, intervals
        cmp [esi], ebx
        jg ending

        dec ecx
    border_start:
        cmp ebx, [esi + 4*ecx] ; проверка, что взятое число >=
следующей левой границы
        jge write_result ; тогда переходим к его записи
        dec ecx

```

```
jmp border_start
```

```
write_result: ; запись числа в массив для вывода
```

```
mov esi, result
```

```
mov ebx, [esi + 4*ecx]
```

```
inc ebx
```

```
mov [edi + 4*ecx], ebx
```

```
ending:
```

```
pop esi
```

```
inc eax
```

```
cmp eax, number_count ; проверка, что весь массив
```

```
чисел обработан
```

```
jne start
```

```
pop edx
```

```
pop ecx
```

```
pop ebx
```

```
pop eax
```

```
pop edi
```

```
pop esi
```

```
ret
```

```
func ENDP
```

```
END
```