

**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 распределения попаданий**  
**псевдослучайных чисел в заданные**  
**интервалы.**

Студент гр.1303

\_\_\_\_\_

Иевлев Е.А.

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.**

#### **Вариант 9**

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.**

Работа содержит два файла: файл на ЯВУ (main.cpp) и файл с ассемблерной вставкой (module1.asm).

В основном файле main.cpp происходит считывание входных данных с дальнейшей проверкой на корректность ввода и передачей в ассемблерный модуль. Массив псевдослучайных чисел генерируется при помощи стандартной библиотеки `ctime` и функции `srand()`. Массив `answer[]` по длине равен количеству интервалов и заполняется значениями-счётчиками для каждого интервала. Оба массива так же передаются в ассемблерный модуль.

После обработки в ассемблерном модуле файлы передаются обратно в main.cpp и выводятся как в файл, так и в консоль.

Обработка чисел в ассемблерном модуле module1.asm:

1. В регистр esx помещается смещение до очередного элемента сгенерированного массива чисел.
2. В массиве интервалов через цикл перебираются все границы и сравниваются с esx
3. В случае попадания числа из esx в интервал увеличиваем соответствующее значение-счётчик интервала в массиве answer. Иначе повторяем пункт 1 для следующего числа

### Тестирование.

Работа программы с заданными условиями представлена на рисунке 1.

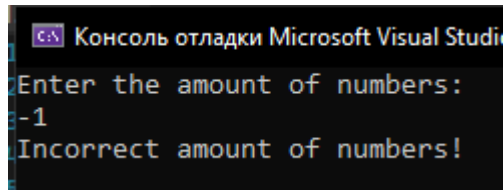
```
Enter the amount of numbers:
6
Enter min value of numbers:
-1
Enter max value of numbers:
5
Enter the amount of intervals:
3
Enter left borders:
-1
0
4
The pseudo-random array is:
1 3 5 4 0 2
N      Borders Numbers amount
1      -1          0
2      0           4
3      4           2
```

Рис. 1

Работа программы при некорректных входных данных:

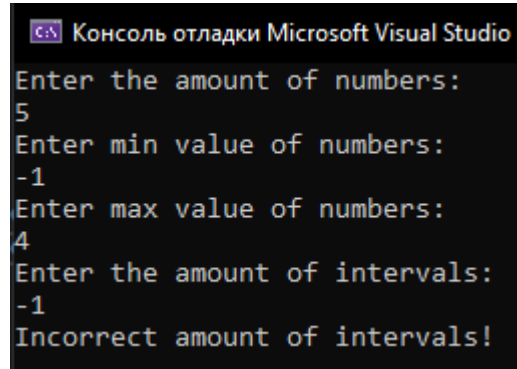
```
Enter the amount of numbers:
6
Enter min value of numbers:
-1
Enter max value of numbers:
5
Enter the amount of intervals:
3
Enter left borders:
-2
The border should be in the [X_min, X_max] interval!
```

Рис. 2



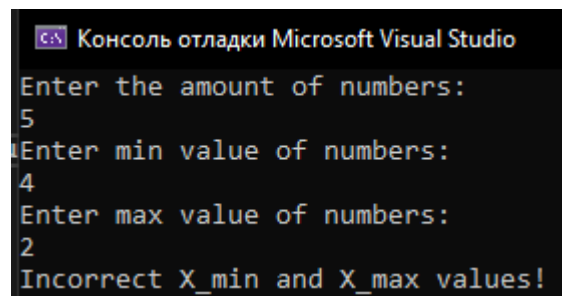
```
Консоль отладки Microsoft Visual Studio
Enter the amount of numbers:
-1
Incorrect amount of numbers!
```

Рис. 3



```
Консоль отладки Microsoft Visual Studio
Enter the amount of numbers:
5
Enter min value of numbers:
-1
Enter max value of numbers:
4
Enter the amount of intervals:
-1
Incorrect amount of intervals!
```

Рис. 4



```
Консоль отладки Microsoft Visual Studio
Enter the amount of numbers:
5
Enter min value of numbers:
4
Enter max value of numbers:
2
Incorrect X_min and X_max values!
```

Рис. 5

### **Выводы.**

В ходе выполнения работы был рассмотрен способ организации связи между ЯВУ и ассемблером, разработана программа, осуществляющая построение частотного распределения попаданий псевдослучайных чисел с равномерным распределением в заданные интервалы.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: *main.cpp*

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <string>

using namespace std;

extern "C" void f(int* inter, int* num, int* answer, int N_int, int N);

int main() {
    srand(static_cast<unsigned int>(time(NULL)));

    int N, N_int, X_min, X_max;

    cout << "Enter the amount of numbers:\n";
    cin >> N;

    if (N <= 0) {
        cout << "Incorrect amount of numbers!\n";
        return 0;
    }

    cout << "Enter min value of numbers:\n";
    cin >> X_min;

    cout << "Enter max value of numbers:\n";
    cin >> X_max;

    if (X_min >= X_max) {
        cout << "Incorrect X_min and X_max values!\n";
        return 0;
    }

    cout << "Enter the amount of intervals:\n";
    cin >> N_int;

    if (N_int <= 0 || N_int > 24) {
        cout << "Incorrect amount of intervals!\n";
        return 0;
    }

    cout << "Enter left borders:" << endl;

    auto inter = new int[N_int + 1];

    for (int i = 0; i < N_int; i++) {
        cin >> inter[i];

        if (inter[i] < X_min || inter[i] > X_max) {
            cout << "The border should be in the [X_min, X_max] interval!\n";
            return 0;
        }
    }

    auto num = new int[N];

    int rand_val = X_max - X_min + 1;
```

```

for (int i = 0; i < N; i++) {
    num[i] = X_min + rand() % rand_val;
}

cout << "The pseudo-random array is:\n";

for (int j = 0; j < N; j++) {
    cout << num[j] << ' ';
}

cout << endl;

auto answer = new int[N_int];

for (int i = 0; i < N_int; i++) {
    answer[i] = 0;
}
f(inter, num, answer, N_int, N);

ofstream file("out.txt");
auto str = "N\tBorders\tNumbers amount";
file << str << endl;
cout << str << endl;
for (int i = 0; i < N_int; i++) {
    auto str_res = to_string(i + 1) + "\t" + to_string(inter[i]) + "\t\t" +
to_string(answer[i]) + "\n";
    file << str_res;
    cout << str_res;
}

return 0;
}

```

Название файла: *module1.asm*

```

.MODEL FLAT, C
.CODE

```

```

f PROC C inter: dword, num: dword, answer: dword, N_int: dword, N: dword

```

```

    push eax
    push ebx
    push ecx
    push edi
    push esi

    mov eax, 0
    mov esi, num

```

```

c_loop:
    mov ebx, 0
    iter:
        cmp ebx, N_int
        jge out_cur_iter
        mov ecx, [esi + 4*eax]
        mov edi, inter
        cmp ecx, [edi + 4*ebx]
        jl out_cur_iter
        inc ebx
        jmp iter

    out_cur_iter:
        dec ebx
        mov edi, answer
        mov ecx, [edi + 4*ebx]
        inc ecx
        mov [edi + 4*ebx], ecx

```

```
        next_number:
            inc eax
            cmp eax, N
            jg exit

    jmp c_loop

exit:
    pop edx
    pop ecx
    pop ebx
    pop eax
    pop edi
    pop esi
    ret
f ENDP

    END
```