

**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.**

Студент гр. 1303

Беззубов Д.В.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

### **Цель работы.**

Рассмотреть способ организации связи ассемблера с ЯВУ на примере связи с языком программирования C++. Разработать программу, выполняющую подсчет попаданий псевдослучайных чисел в заданные интервалы.

### **Задание.**

На языке C программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND\_GEN (при его отсутствии получить у преподавателя).

Следует привести числа к целому виду с учетом диапазона изменения.

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.**

В ходе выполнения работы, написаны два исходных файла – основное тело программы, описанное в файле *main.cpp*, и модуль, выполняющий обработку данных, написанный на языке ассемблера.

В функции *main()* происходит считывание входных данных, а так же проверка ввода на корректность. Так же в данной функции происходит подготовка данных для передачи их в ассемблерный модуль. С помощью

функции стандартной библиотеки `std::normal_distribution<>()` генерируется массив псевдослучайных чисел с нормальным распределением Гаусса.

Затем все сгенерированные данные передаются в ассемблерный блок. Полученные результаты выводятся на экран и в файл средствами ЯВУ.

В ассемблерном модуле обработка осуществляется следующим образом:

1. В регистр EBX помещается смещение до очередного числа из сгенерированного массива
2. Циклом перебираем левые границы интервалов, начиная со второго (по условию, первый левый интервал не меньше минимального значения, поэтому, в случае, если число меньше второй левой границы – оно попадает в первый интервал).
3. В случае, если число попадает в какой-то из интервалов – мы сохраняем значение регистра EBX на стек, затем в него сохраняем смещение до элемента массива, соответствующего нужной левой границе и увеличиваем это значение.
4. В ином случае, просто переходим к следующему числу.

### Тестирование.

На рисунках представлены результаты тестирования программы:

```
Input count of numbers:
1000
Input min value of numbers:
-1
Input max value of numbers:
2
Input count of intervals:
3
Input left borders:
-1 0 1
input right border:
2

N      Borders Numbers` count
1      -1          167
2       0          707
3       1          126
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1 – Результат работы при правильном вводе

Проверка работы программы при некорректных значениях границ интервалов, кол-ва интервалов и кол-ва генерируемых чисел:

```
Input count of numbers:
-1
Input min value of numbers:
-1
Input max value of numbers:
1
Input count of intervals:
3
incorrect count of numbers
```

Рисунок 2 – некорректное кол-во чисел

```
Input count of numbers:
10
Input min value of numbers:
-12
Input max value of numbers:
-13
Input count of intervals:
3
incorrect X_min and X_max values
```

Рисунок 3 – введенное максимальное значение меньше минимального

```
Input count of numbers:
10
Input min value of numbers:
-1
Input max value of numbers:
1
Input count of intervals:
3
Input left borders:
-1 1 0
input right border:
2
incorrect borders
```

Рисунок 4 – некорректные границы интервалов

### **Вывод.**

Рассмотрен способ организации связи ассемблера с ЯВУ. Разработана программа, строящая частотное распределение попадания псевдослучайных чисел, сгенерированных с нормальным распределением, 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 N_int, int N, int* numbers,
int* final_answer);

int main() {

    int N, X_min, X_max, N_int;

    cout << "Input count of numbers:\n";
    cin >> N;
    cout << "Input min value of numbers:\n";
    cin >> X_min;
    cout << "Input max value of numbers:\n";
    cin >> X_max;
    cout << "Input count of intervals:\n";
    cin >> N_int;

    if (N <= 0) {
        cout << "incorrect count of numbers\n";
        return 0;
    }

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

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

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

    auto intervals = new int[N_int + 1];

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

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

```

    }
}

cout << "input right border:\n";
cin >> intervals[N_int];

for (int i = 0; i < N_int-1; i++) {
    for (int j = i+1; j < N_int; j++) {
        if (intervals[j] < intervals[i]) {
            cout << "incorrect borders\n";
            return 0;
        }
    }
}

auto numbers = new int[N];
random_device rd;
mt19937 generator(rd());
normal_distribution<> dist((X_max+X_min)/2);

int i = 0;

while(i < N){
    double curr = dist(generator);
    if (curr >= X_min && curr <= X_max) {
        numbers[i] = int(curr);
        i++;
    }
}

cout << endl;

auto final_answer = new int[N_int];

for (int i = 0; i < N_int; i++) {
    final_answer[i] = 0;
}
func(intervals, N_int, N, numbers, final_answer);

ofstream file("output.txt");
auto str = "N\tBorders\tNumbers` count";
file << str << endl;
cout << str << endl;
for (int i = 0; i < N_int; i++) {
    auto str_res = to_string(i + 1) + "\t" + to_string(intervals[i])
+ "\t\t" + to_string(final_answer[i]) + "\n";
    file << str_res;
    cout << str_res;
}
system("Pause");
return 0;
}

```

## Название файла: module.asm

```
.MODEL FLAT, C
.CODE

func PROC C intervals: dword, N_int: dword, N: dword, numbers: dword,
final_answer: dword

    push eax
    push ebx
    push ecx
    push edi
    push esi

    mov esi, numbers
    mov edi, final_answer
    mov eax, 0

checking_loop:

    mov ebx, [esi+4*eax]

    push esi
    mov ecx, 1

    mov esi, intervals

new_inter:
    cmp ebx, [esi + 4*ecx]
    jge next_inter

    sub ecx, 1

    push ebx
    mov ebx, [edi + 4*ecx]
    inc ebx
    mov [edi + 4*ecx], ebx
    pop ebx

    inc eax
    cmp eax, N
    pop esi
    jl checking_loop
    jmp exit

next_inter:
    inc ecx
    cmp ecx, N_int
    jle new_inter
    inc eax
```

```
    pop esi
    jmp checking_loop

exit:
    pop edx
    pop ecx
    pop ebx
    pop eax
    pop edi
    pop esi
ret
func ENDP
END
```