

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

ОТЧЕТ
по лабораторной работе №5
по дисциплине «Организация ЭВМ и систем»
Тема: Организация связи Ассемблера с ЯВУ на примере
программы построения частотного распределения попаданий
псевдослучайных целых чисел в заданные интервалы

Студент гр. 1303

Чубан 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 3 данных для передачи их в ассемблерный модуль. С помощью функции стандартной библиотеки `std::uniform_int_distribution<int>`

генерируется массив псевдослучайных чисел с нормальным распределением Гаусса. Затем все сгенерированные числа передаются в ассемблерный блок. Полученные результаты выводятся на экран и в файл средствами ЯВУ.

В ассемблерном модуле обработка осуществляется следующим образом:

1. В регистр ECX помещается смещение до очередного числа из сгенерированного массива
2. Циклом перебираем левые границы интервалов и сравниваем его с числом, помещенным в ECX.
3. В случае, если число попадает в какой-то из интервалов – увеличивается значение в массиве `final_answer[]`, соответствующее данному интервалу.
4. В ином случае, просто переходим к следующему числу.

Тестирование.

```
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 0 1
0 -1 0 0 -1 -1 0 -1 0 -1
N      Borders      Numbers` count
1      -1           5
2      0            5
3      1            0
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1 – Результат работы при правильном вводе

```
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 количество чисел

```
Input count of numbers:
100
Input min value of numbers:
10
Input max value of numbers:
-1
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 0 -1
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++;
    }
}

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

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, 0
```

```
    iter:
```

```
        cmp ebx, N_int
```

```
        jge out_cur_iter
```

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

```
        mov edi, intervals
```

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

```
        jnl out_cur_iter
```

```
        inc ebx
```

```
        jmp iter
```

```
out_cur_iter:
```

```
    dec ebx
```

```
    mov edi, final_answer
```

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

```
    inc ecx
```

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



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

    jmp checking_loop

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