**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

* + 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.

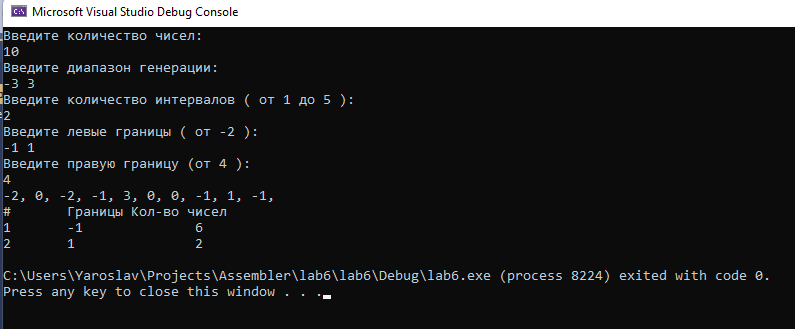
После вызова ассемблерного блока выводится результат его выполнения.

Module.asm

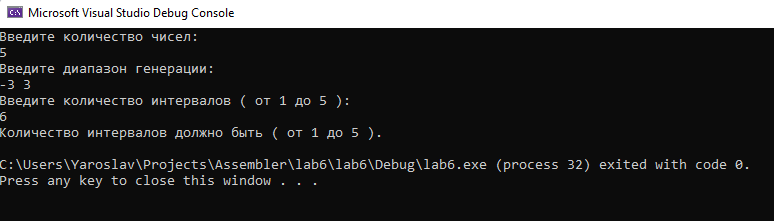
* + блоке происходит поиск количества чисел, вошедших в каждый заданный интервал. Поиск реализован в виде двойного цикла. В первом происходит последовательный выбор из массива сгенерированных чисел. В во втором - перебор левых границ для нахождения нужного диапазона. Перед вторым циклом выбранное число 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.**

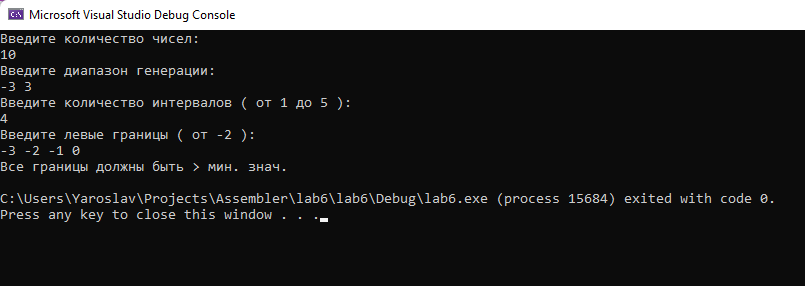
Тест 1.



Тест 2.



Тест 3.



**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

8