МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА(ЛЕНИНА)

КАФЕДРА МО ЭВМ

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.

Студентка гр. 1381 Деркачева Д.Я.

Преподаватель Ефремов М. А.

Санкт-Петербург

2022

Цель работы

Изучить организацию связи ассемблера с языками высокого уровня,в нашем случае c++ .

Текст задания

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение.

Далее должен вызываться ассемблерный модуль(модули) для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину.

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.

Исходные данные:

1. Длина массива псевдослучайных целых чисел - NumRanDat (<= 16K, K=1024)

2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax] , значения могут быть биполярные;

3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24)

4. Массив левых границ интервалов разбиения LGrInt (должны принадлежать интервалу [Xmin, Xmax]).

Результаты:

1. Текстовый файл, строка которого содержит: - номер интервала, - левую границу интервала, - количество псевдослучайных чисел, попавших в интервал. Количество строк равно числу интервалов разбиения.

2. График, отражающий распределение чисел по интервалам. (необязательный результат)

Ход выполнения работы

Вариант 2- подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде двух ассемблерных модулей, первый из которых формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат. Это распределение должно выводиться в текстовом виде для контроля. Затем вызывается второй ассемблерный модуль, который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами). Это распределение возвращается в головную программу и выдается как основной результат в виде текстового файла и, возможно, графика.

Main - Совершается считывание необходимых значений с выводом ошибок, если значения противоречат заданию. Вместе с этим есть возможность ввода левых границ интервалов в произвольном порядке благодаря функции быстрой сортировки c++ полученного массива значений. Далее написана часть с реализацией генерации псевдослучайных чисел с нормальным распределением

при помощи mersenne twister. Далее последовательно вызываются ассемблерные процедуры (Module1, Module2) с указанием всех требуемых, для обработки в каждой из процедур, значений. После завершения обработки поступивших значений – в файле «result.txt» и на экране демонстрируются результаты работы в виде таблиц частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

Module1 — Выполняет распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ. В процедуру из исходной программы поступают: Массив псевдослучайных чисел, количество этих чисел, наименьший элемент этого массива и результирующий массив. Внутри процедуры посредством цикла по числу всех элементов в массиве реализован счёт количества чисел, входящих в конкретный единичный интервал.

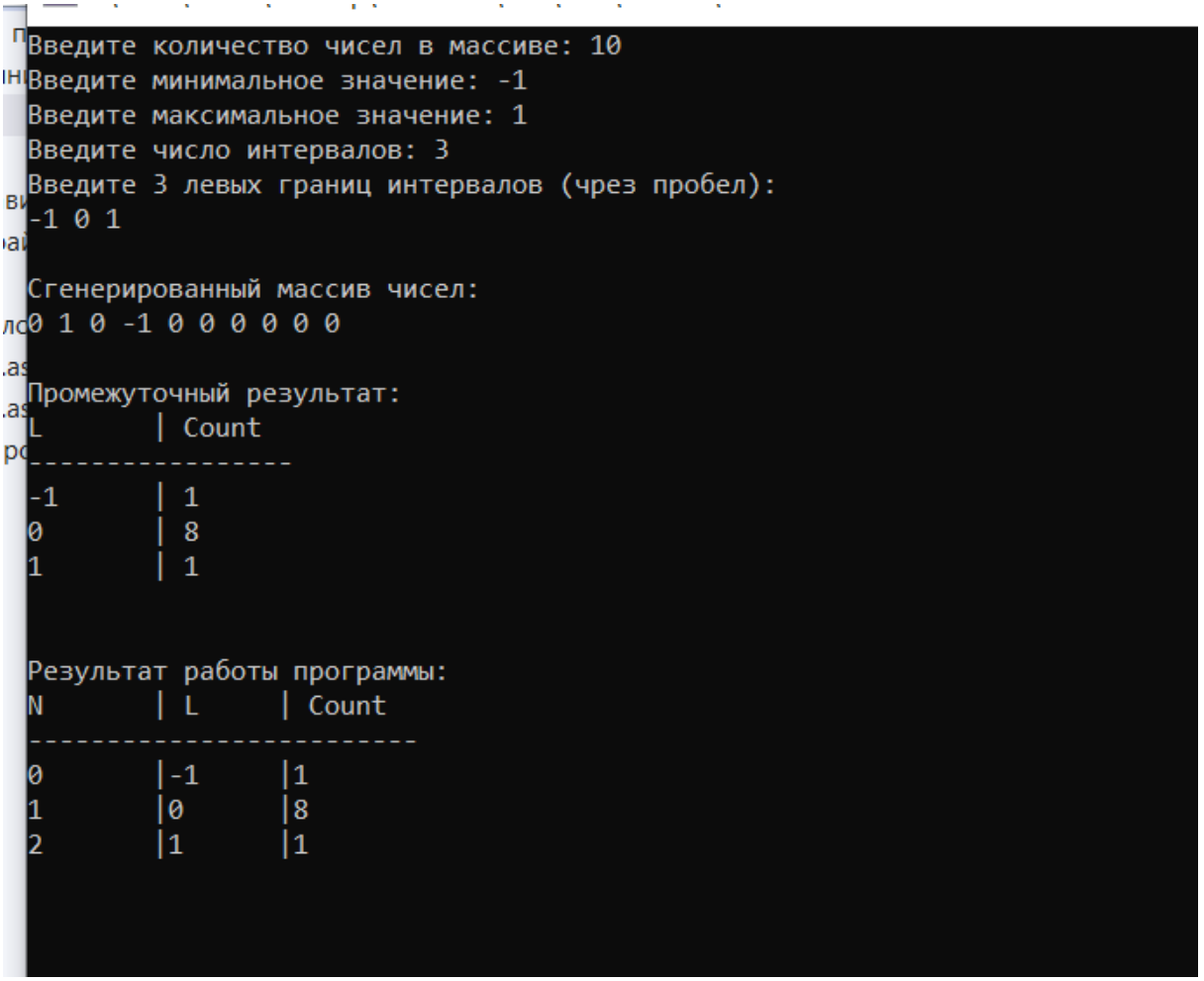
Module2 — Выполняет распределение исходных чисел по интервалам, на основе результатов из ранее вызванного ассемблерного модуля. В процедуру из исходной программы поступают: Массив распределения псевдослучайных чисел по единичным интервалам, левые границы заданных интервалов, число этих интервалов, наибольшее и наименьшее значения из случайных чисел и массив для хранения финального результата распределения. Изначально определяется значение левой границы первого интервала, все числа после этой границы и до следующей– обрабатываются: к результирующему массиву (который, на данном шаге, хранит количество встреч чисел до текущего в пределах одного интервала) добавляется количество встреч текущего. Затем переходим к следующему псевдо случайному числу. Этот процесс повторяется до последнего элемента, обходя все заданные интервалы.

Текст исходного файла программы

Текст исходной программы lab6.cpp, module1.asm, module2.asm см. в приложении А.

Тестирование

Рис.1. Результат тестирования



Выводы по работе

В ходе выполнения лабораторной работы была разработана программа , которая дает возможность разобраться в организации связи Ассемблера с ЯВУ. Также было реализовано построение частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы .

Приложение А

Текст исходного файла lab6.cpp

```
#include <iostream>

#include <fstream>

#include <random>

#include <string>


extern "C" void module1(int* arr, int n, int* res1, int min);

extern "C" void module2(int* distr, int* interv, int min, int max,
int* res2);


using namespace std;




int comp(const void* a, const void* b) {

     return (*(int*)a - *(int*)b);

}


int main() {

     setlocale(LC_ALL, "Russian");

     int n;

     int min;

     int max;

     int NInt;
```

```
    cout << "Введите количество чисел в массиве: ";
    cin >> n;
    if (n > 16000 || n <= 0) {
        cout << "Неккоректно заданный объем массива, попробуйте еще раз\n";
        return 0;
    }
    cout << "Введите минимальное значение: ";
    cin >> min;
    cout << "Введите максимальное значение: ";
    cin >> max;
    int D = max - min;
    if (D <= 0) {
        cout << "Минимальное значение не может быть больше максимального, попробуйте еще раз\n";
        return 0;
    }
    cout << "Введите число интервалов: ";
    cin >> NInt;
    if (NInt >= 24 || NInt < 1 || NInt < D + 1) {
        cout << "Неверное число интервалов, попробуйте еще раз\n";
        return 0;
    }


    int* interv = new int[NInt + 1];
    int* result_modul2 = new int[n];
    cout << "Введите " << NInt << " левых границ интервалов (чрез пробел):\n";
```

```
	for (int i = 0; i < NInt; i++) {
		cin >> interv[i];
		result_modul2[i] = 0;
	}
	qsort(interv, NInt, sizeof(int*), comp);

	if (interv[NInt] < min || interv[NInt] > max) {
		cout << "Левые границы не входят в диапозон значений, попробуйте еще раз\n";
		return 0;
	}

	random_device rd;
	mt19937 gen(rd());
	float l = float(max + min) / 2;
	float r = float(max - min) / 4;
	normal_distribution<float> conc_gen(l, r);

	interv[NInt] = max + 1;
	int* result_module1 = new int[abs(D) + 1];
	int* arr = new int[n];

	for (int i = 0; i < abs(D) + 1; i++) {
		result_module1[i] = 0;
	}

	cout << "\nСгенерированный массив чисел:" << endl;
```

```
    for (int i = 0; i < n; i++) {
        arr[i] = int(conc_gen(gen));
        cout << arr[i] << ' ';
    }
    cout << endl;

    module1(arr, n, result_module1, min);

    ofstream file("result.txt");
    string inter_res = "Промежуточный результат:\n";
    string inter_table = "L\t| Count";

    file << inter_res << inter_table << endl;
    cout << "\n" << inter_res << inter_table << endl;
    cout << "-----------------\n";
    for (int i = 0; i < abs(D) + 1; i++) {
        string res1 = (to_string(min + i) + "\t| " +
to_string(result_module1[i]) + "\n");
        file << res1;
        cout << res1;
    }

    module2(result_module1, interv, min, max, result_modul2);

    string final_res = "\nРезультат работы программы:\n";
    string final_table = "N\t| L\t| Count";
```

```
	file << final_res << final_table << endl;

	cout << "\n" << final_res << final_table << endl;

	cout << "-------------------------\n";

	for (int i = 0; i < NInt; i++) {

		string	res2	=	(to_string(i)	+	"\t|"	+
to_string(interv[i]) + "\t|" + to_string(result_modul2[i]) + "\n");

		file << res2;

		cout << res2;

	}


	return 0;

}
```

Текст исходного файла module1.asm

```
.586

.MODEL FLAT, C

.CODE

PUBLIC C module1

module1 PROC C arr: dword, n: dword, res: dword, min: dword


push esi

push edi

mov esi, res

mov edi, arr

mov ecx, n


lp:
```

```
    mov eax,[edi]
    sub eax,min
    mov ebx,[esi+4*eax]
    add ebx, 1
    mov [esi+4*eax],ebx
    add edi,4
    loop lp

pop edi
pop esi
ret
module1 ENDP
END
```

Текст исходного файла module1.asm

```
.586
.MODEL FLAT, C
.CODE
PUBLIC C module2
module2 PROC C distr: dword, interv: dword, min: dword, max: dword,
res: dword

push esi
push edi
push eax
push ebx
```

```
push ecx

mov esi, res
mov edi, interv
mov eax, min
mov ebx, 0
mov ecx, 0


Start:
cmp eax, [edi+4*ebx]
jl Act
add ebx, 1
jmp Start

Act:
     push edi
     push eax
     mov edi, distr
     sub ebx, 1
     mov eax, [esi+4*ebx]
     mov edx, [edi+4*ecx]
     add eax, edx
     mov [esi+4*ebx], eax
     pop eax
     pop edi
          push ecx
```

```
            mov ecx, max
            cmp eax, ecx
            pop ecx
            je final
      add ecx, 1
      add eax, 1
      jmp Start

final:
      pop eax
      pop ebx
      pop ecx
      pop edi
      pop esi
      ret
module2 ENDP
END
```