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
Вариант 2

Студентка гр.1381 _____ Рымарь М.И.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Научиться связывать язык Ассемблера и язык высокого уровня так, чтобы функции ассемблерного модуля вызывались из программы на C++.

Задание.

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND_GEN (при его отсутствии программу датчика получить у преподавателя). 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)

Для бригад с четным номером: 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 выдаётся как основной результат в виде текстового файла и, возможно, графика.

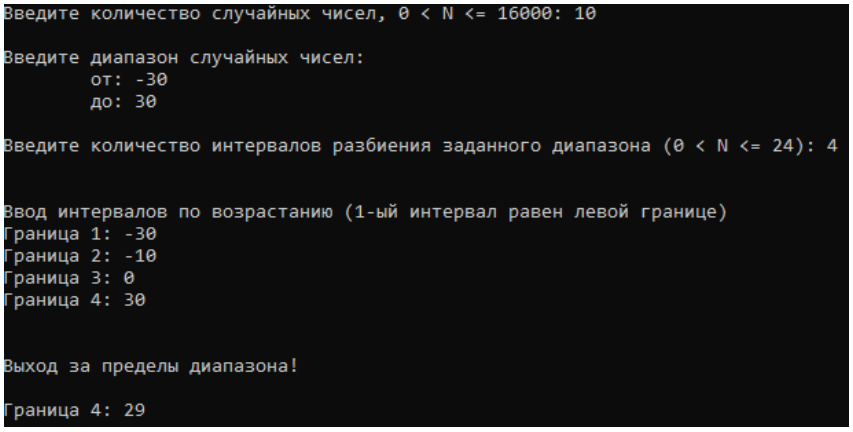
Выполнение работы.

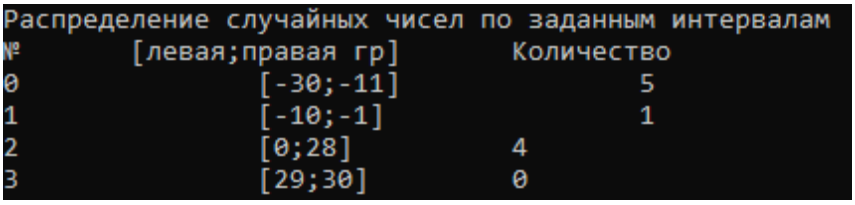
Для реализации поставленной задачи написана программа с использованием высокоуровневого языка C++. В качестве интегрируемой среды разработки использовалась Visual Studio 2022. В файле *lab6.cpp* реализовано считывание начальных данных. Левые границы заносятся в массив *ArrLeftBorder*, генерируемые числа добавляются в массив *ArrNumber*. Отдельно создаётся массив, который будет хранить результат работы. В ассемблерный модуль в *func1* – распределение по единичным интервалам – передаётся указатель на массив чисел, количество границ, массив для интервалов единичной границы и минимальное значение границы. Это распределение выводится в текстовом виде в консоль для контроля. Также по завершении работы программы распределение записывается в текстовый файл «result.txt». В ассемблерный модуль в func2 – распределение по определённым интервалам – передаётся указатель на массив с единичным интервалом, указатель на массив с

единичными границами, указатель на массив для интервалов заданной длины, количество левых границ и минимальное значение левой границы. По результатам прошлого промежуточного распределения формируется окончательное распределение псевдослучайных чисел по интервалам произвольной длины. Также по завершении работы программы распределение записывается в текстовый файл «result.txt».

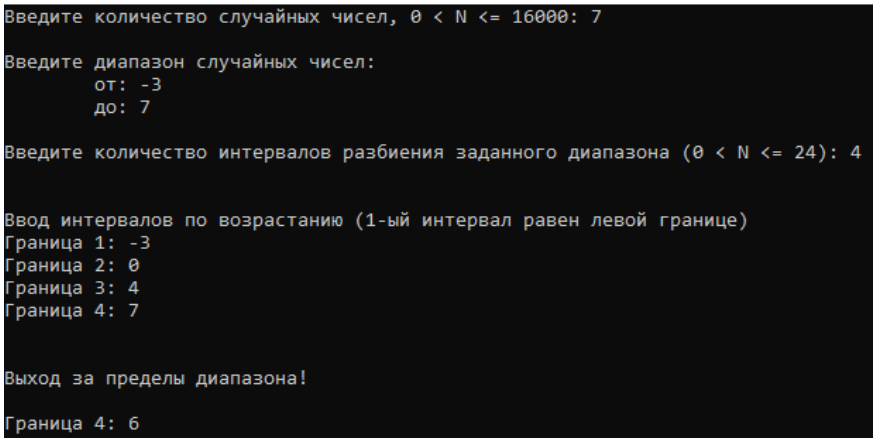
Тестирование.

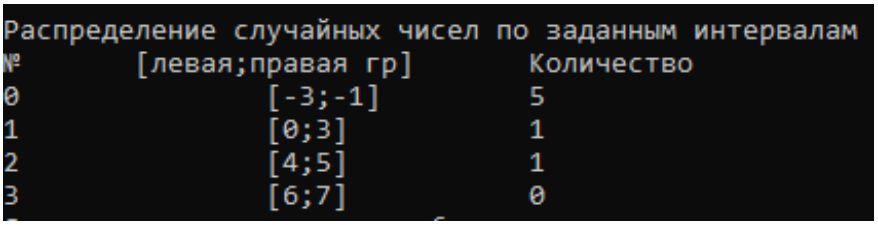
Для тестирования написанной программы было использовано три теста. Тест 1 приведён на рисунках 1 и 2. Тест 2 приведён на рисунках 3 и 4. Тест 3 приведён на рисунках 5 и 6.



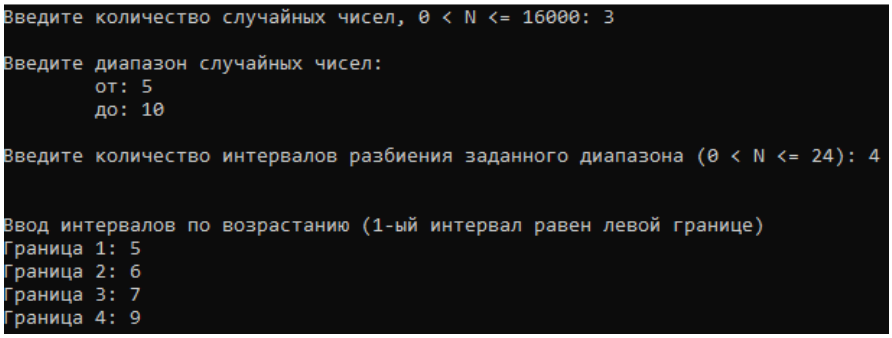


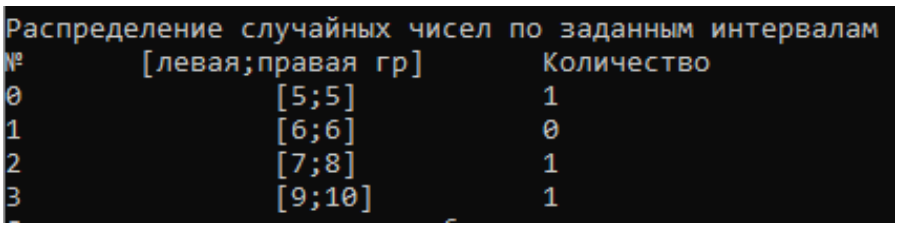
Рисунки 1, 2 – Первый тест





Рисунки 3, 4 – Второй тест





Рисунки 5, 6 – Третий тест

Выводы.

В ходе выполнения лабораторной работы была написана программа на языке Ассемблера, которая строит частотное распределение попаданий псевдослучайных чисел в заданные интервалы. В результате написания работы были получены практические навыки связи программирования на ЯВУ и Ассемблере.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Название файла: *lab6.cpp*

```
#include <iostream>
#include <random>
#include <fstream>
#include <ctime>
using namespace std;

#define NUMBER  16000      //максимальное длина числа
#define BORD  24      //максимальное количество интервалов
//void sort(int* &ArrNumber, int len);
void get_arr(int& QuantNumber, int*& ArrNumber, int& Xmin, int& Xmax, int&
QuantBorder, int*& ArrLeftBorder);
void generation(int*& ArrNumber, int QuantNumber, int min, int max);
void res_func1(int Xmin, int* ArrNumber, int QuantNumber);
void  res_func2(int  Xmax,  int*  ArrLeftBorder,  int*  ArrNumber,  int
QuantNumber);
ofstream fout("result.txt");

extern "C"
{
     void  func1(int  ArrNumber[],  int  QuantNumber,  int  ArrInter1[],  int
Xmin);
     void   func2(int   ArrWith1Range[],   int   ArrLeftBorder[],   int
ArrInterDif[], int QuantBorder, int Xmin);
}

int main(void) {
     setlocale(LC_ALL, "rus");

     int QuantNumber = 0; //кол-во псевдослучайных чисел
     int  Xmin = 0, Xmax = 0, MainRange = 0;
     int  QuantBorder = 0; //кол - во границ
     int* ArrNumber = NULL;          //массив чисел
     int* ArrLeftBorder = NULL;       //массив левых границ
     int* ArrRightBorder = NULL; //массив правых границ
     int* ArrInter1 = NULL;     //для интервалов единичной длины
     int* ArrInterDif = NULL;   //для интервалов заданной длины
```

```
	get_arr(QuantNumber, ArrNumber, Xmin, Xmax, QuantBorder, ArrLeftBorder); // получение псевдослучайных чисел

	ArrRightBorder = new int[QuantBorder]; // создание массива правых границ, исходя из массива левых границ и граничного значения Xmax
	for (int i = 0; i < QuantBorder - 1; i++)
		ArrRightBorder[i] = ArrLeftBorder[i + 1] - 1;

	ArrRightBorder[QuantBorder - 1] = Xmax;

	MainRange = Xmax - Xmin + 1;
	ArrInter1 = new int[MainRange] {0};
	ArrInterDif = new int[QuantBorder] {0};
	func1(ArrNumber, QuantNumber, ArrInter1, Xmin); //распределение по ед. интервалам
	func2(ArrInter1, ArrRightBorder, ArrInterDif, QuantBorder, Xmin); //распределение по опр. интервалам
	res_func1(Xmin, ArrInter1, MainRange);	//вывод результата первой процедуры
	res_func2(Xmax, ArrLeftBorder, ArrInterDif, QuantBorder);  //вывод результата второй процедуры

	system("pause");
	delete[] ArrInterDif;
	delete[] ArrInter1;
	delete[] ArrNumber;
	delete[] ArrLeftBorder;
	delete[] ArrRightBorder;
	return 0;
}

void get_arr(int& QuantNumber, int*& ArrNumber, int& Xmin, int& Xmax, int& QuantBorder, int*& ArrLeftBorder)
{

	do {
		cout << "Введите количество случайных чисел, 0 < N <= " << NUMBER << ": ";
		cin >> QuantNumber;
		if (QuantNumber <= 0 || QuantNumber > NUMBER)
			cout << "\nОшибка диапазона!\n\n";
	} while (QuantNumber <= 0 || QuantNumber > NUMBER);
	ArrNumber = new int[QuantNumber];
	do {
```

```
            cout << "\nВведите диапазон случайных чисел: \n" << " от:    ";
cin >> Xmin;
            cout << "  до: ";  cin >> Xmax;
            if (Xmax <= Xmin)
                  cout    <<    "\nНеверное    задание    границ!    Повторите
попытку.\n\n";
      } while (Xmax <= Xmin);
      generation(ArrNumber, QuantNumber, Xmin, Xmax);

      do {
            cout << "\nВведите количество интервалов разбиения заданного
диапазона (0 < N <= " << BORD << "): ";
            cin >> QuantBorder; cout << endl;
            if (QuantBorder <= 0 || QuantBorder > BORD)
                  cout << "\nОшибка: количество интервалов не входит в
указанный диапазон!Повторите попытку.\n";
      } while (QuantBorder <= 0 || QuantBorder > BORD);

      ArrLeftBorder = new int[QuantBorder];
      cout << "\nВвод интервалов по возрастанию (1-ый интервал равен левой
границе)\n";
      ArrLeftBorder[0] = Xmin;   //левая граница
      cout << "Граница 1: " << Xmin << "\n";
      int tmp = 0;
      for (int i = 1; i < QuantBorder; i++)
      {
            do {
                  cout << "Граница " << i + 1 << ": ";
                  cin >> tmp;
                  if (tmp <= ArrLeftBorder[i - 1] || tmp >= Xmax)
                  {
                        cout << "\n\nВыход за пределы диапазона!\n\n";
                  }
                  else
                  {
                        ArrLeftBorder[i] = tmp;
                        break;
                  }
            } while (true);
      }
}

void generation(int*& ArrNumber, int len, int min, int max)
{
```

```
	random_device rd; // класс, кот. описывает результаты ,равномерно распределенные в замкнутом диапазоне [ 0, 2^32).
	mt19937 gen(rd());
	uniform_int_distribution<> distr(min, max); //формирует равномерное распределение целых чисел в заданном интервале
	for (int i = 0; i < len; i++) {
		ArrNumber[i] = distr(gen);
	}
}

void res_func1(int Xmin, int* ArrInter1, int QuantNumber)
{
	cout << "\nРаспределение случайных чисел по интервалам единичной длины\n"; fout << "\nРаспределение случайных чисел по интервалам единичной длины\n";
	cout << "Число\tКол-во\n"; fout << "№\tЧисло\tКол-во\n";


	for (int i = 0; i < QuantNumber; i++) {
		cout << Xmin + i << "\t" << ArrInter1[i] << '\n'; fout << i + 1 << '\t' << Xmin + i << "\t" << ArrInter1[i] << '\n';
	}
}


void res_func2(int Xmax, int* ArrLeftBorder, int* ArrInterDif, int QuantNumber)
{

	cout << "\nРаспределение случайных чисел по заданным интервалам\n"; fout << "\nРаспределение случайных чисел по заданным интервалам\n";
	cout << "№\t[левая;правая гр]\tКоличество\t\n"; fout << "№\t[левая;правая гр]\tКоличество\t\n";
	for (int i = 0; i < QuantNumber; i++)
	{
		cout << i << "\t\t[" << ArrLeftBorder[i] << ";";
		fout << i << "\t\t[" << ArrLeftBorder[i] << ";";
		if (i == QuantNumber - 1)
		{
			cout << Xmax << "]"; fout << Xmax << "]";
		}
		else
		{
			cout << ArrLeftBorder[i + 1] - 1 << "]";
```

```
                fout << ArrLeftBorder[i + 1] - 1 << "]";
            }

            cout << "\t\t" << ArrInterDif[i] << '\n'; fout << "\t\t" <<
ArrInterDif[i] << '\n';

        }
}
```

Название файла: *lab6asm.asm*

```
.586p
.MODEL FLAT, C
.CODE

PUBLIC C func1
func1 PROC C USES EDI ESI,\
ArrNumber:DWORD, QuantNumber:DWORD, ArrInter1:DWORD, Xmin:DWORD

MOV EDI, ArrNumber                                  ;Адрес массива
случайных чисел
MOV ESI, ArrInter1                                 ;Адрес массива счетчика
чисел
MOV ECX, QuantNumber                               ;Длина массива случайных
чисел
MOV EAX, Xmin

CYCLE:
                                                   MOV EBX, [EDI]
;Извлечение случайного числа N
                                                   SUB EBX, EAX
;Вычитание левой границы диапазона
                                                   ADD DWORD PTR[ESI+4*EBX], 1;
;Увеличение счетчика числа на 1
                                                   ADD EDI, 4
;Переход к следующему числу
LOOP CYCLE

RET
func1 ENDP



PUBLIC C func2
func2 PROC C USES EDI ESI,\
ArrInter1:DWORD, ArrRightBorder:DWORD, ArrInterDif:DWORD,\
QuantBorder:DWORD, Xmin:DWORD

MOV EDI, ArrRightBorder     ;Адрес массива правых границ
MOV ESI, ArrInter1      ;Адрес массива счетчика чисел
MOV EAX, ArrInterDif    ;Адрес массива заданных интервалов
MOV ECX, QuantBorder     ;Количество разбиений (интервалов)
MOV EBX, XMIN


XOR EDX, EDX
```

```
CYCLE:
                                        CMP EBX, [EDI]
                                        JG NEXT_RANGE      ;Переход,
если число больше текущей границы
                                        ADD EDX, [ESI]
;Накопление
                                        INC EBX               ;Переход к
следующему числу
                                        ADD ESI, 4            ;Переход к
следующему элементу распр. чисел с единичным диапазоном
                                        JMP CYCLE
NEXT_RANGE:                                          ;Достигнута правая
граница интервала
                                        MOV [EAX], EDX        ;Помещаем
в массив с заданным распр. накопленное значение
                                        XOR EDX, EDX          ;Обнуляем
значение
                                        ADD EAX, 4             ;Переход
к следующем элементу массива
                                        ADD EDI, 4            ;Переход к
следующей границе
LOOP CYCLE

RET
func2 ENDP
END
```