Отчет

по лабораторной работе №6

по дисциплине “Архитектура компьютера”

Тема: “Организация связи Ассемблера с ЯВУ на примере программы

построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы”

Вариант №2

Студент гр. 5383 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Допира В. Е.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кирьянчиков В.А.

**Цель работы**

На языке высокого уровня (Pascal или С) программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное или гауссовское (ограниченное интервалом [Mx-3\*S, Mx+3\*S]) распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND\_GEN (пpи 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.

Исходные данные:

1. Длина массива псевдослучайных целыхчисел - NumRanDat (<= 16K)

2. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax] (м.б. биполярный, например, [-100, 100])

3. Массив псевдослучайных целых чисел.

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

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

В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину.

Результаты:

1. Обязательный - текстовая таблица, строка которой содержит:

- номер интервала,

- левую границу интервала,

- количество псевдослучайных чисел, попавших в интервал.

Количество строк должно быть равно числу интервалов разбиения. Таблица должна выводиться на экран и сохраняться в файле.

2. Необязательный - график, отражающий распределение чисел по интервалам.

Задание:

2. Для бригад с четным номером: программа формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде двух ассемблерных модулей (процедур), первый из которых формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат. Также это распределение выводится на экран и сохраняется в файле. 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.

Замечание: На ЯВУ следует реализовать только ввод (возможно с контролем), вывод и генерацию псевдослучайных целых чисел. Всю остальную функциональность следует программировать на ассемблере.

**Код программы**

**Файл на с++:**

#include <iostream>

#include <fstream>

#include <locale>

#include <string.h>

#include <stdint.h>

#include <cstdlib>

#include <ctime>

#include <list>

#include <ctype.h>

#include <random>

#include <ratio>

#include <chrono>

#include <cinttypes>

using namespace std;

extern "C" {

void first(int NumRanDat, int\* RanDat, int Xmin, int\* fInt);

void second(int\* LGrInt, int NInt, int\* fInt, int fIntSize, int Xmax, int\* Freqs);

}

int compareInt32\_t (const void \* a, const void \* b)

{ return ( \*(int32\_t\*)a - \*(int32\_t\*)b );}

int main(int argc, char \*argv[])

{

setlocale(LC\_ALL, "Russian");

ofstream fout("output.txt");

istream& in = cin;

cout << "Введите количество чисел NumRanDat (>0, <=16000)" << endl;

int NumRanDat;

in >> NumRanDat;

while (in.fail() || NumRanDat <= 0 || NumRanDat > 16000) {

in.clear();

in.ignore(INT\_MAX, '\n');

cout << "NumRanDat не принадлежит промежутку допустимых значений, повторите ввод." << endl;

in >> NumRanDat;

}

cout << "Введите интервал случайных чисел[Xmin, Xmax]" << endl;

int Xmin, Xmax;

in >> Xmin >> Xmax;

while (in.fail() || Xmin >= Xmax) {

in.clear();

in.ignore(INT\_MAX, '\n');

cout << "Неправильно указаны граничные значения, повторите ввод." << endl;

in >> Xmin >> Xmax;

}

cout << "Введите количество промежутков разбиения NInt (>0, <=24)" << endl;

int NInt;

in >> NInt;

while (in.fail() || NInt <= 0 || NInt > 24) {

in.clear();

in.ignore(INT\_MAX, '\n');

cout << "NInt не принадлежит промежутку допустимых значений, повторите ввод" << endl;

in >> NInt;

}

++NInt;

int LGrInt[25];

LGrInt[0]=Xmin;

for (int i = 1; i < NInt; ++i) {

cout << "Введите левую границу LGrInt[" << i <<"], должна принадлежать ["<<Xmin<<","<<Xmax<<"]" << endl;

in >> LGrInt[i];

while (in.fail() || LGrInt[i] < Xmin || LGrInt[i] > Xmax) {

in.clear();

in.ignore(INT\_MAX, '\n');

cout << "Неверно указана граница, повторите ввод" << endl;

in >> LGrInt[i];

}

}

int RanDat[16000] = {0};

mt19937 rng(chrono::system\_clock::now().time\_since\_epoch().count()); //генератор случайных чисел

uniform\_int\_distribution<int> dist(Xmin, Xmax);

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

RanDat[i] = dist(rng);

}

int fIntSize = Xmax-Xmin+1; // размер интервала

int\* fInt = new int[fIntSize]();

cout << 4 << endl;

first(NumRanDat,RanDat,Xmin,fInt);

int Freqs[25];

second(LGrInt, NInt, fInt, fIntSize, Xmax, Freqs);

ofstream fin("out.txt");

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

{

cout <<"Интервал"<< i << " " <<LGrInt[i] << ' ' << Freqs[i] << endl;

fin <<"Интервал"<< i << " " <<LGrInt[i] << ' ' << Freqs[i] << endl;

}

fin.close();

return 0;

}

**count-32.s**

.intel\_syntax noprefix

.global \_first

# void first(int NumRanDat, int\* RanDat, int Xmin, int\* fInt);

\_first:

push ebp

mov ebp, esp

push eax

push ebx

push ecx

push esi

push edi

mov ecx, [ebp+8] #NumRanDat

mov esi, [ebp+12] #Randat

mov edi, [ebp+20] #Xmin

cycle:

cmp ecx, 0

je exit

dec ecx

mov eax, [esi+ecx\*4] #от RanDat-1 до 0

sub eax, [ebp+16]

inc dword ptr [edi+eax\*4]

jmp cycle

exit:

pop edi

pop esi

pop ecx

pop ebx

pop eax

pop ebp

ret

.global \_second

#void second(int\* LGrInt, int NInt, int\* fInt, int fIntSize, int Xmax, int\* Freqs);

\_second:

push ebp

mov ebp, esp

sub esp, 8

push eax

push ebx

push ecx

push edx

push esi

push edi

mov eax, [ebp+24] #Xmax

mov [ebp-4], eax #local varuable (текущее число, которое мы проверяем на принадлежность интервалу)

mov esi, [ebp+16] # fInt

mov edi, [ebp+8] # LgrInt

mov ebx, [ebp+12] # Nint

mov edx, [ebp+20] # fIntSize

dec edx

checklgrint:

dec ebx

cmp ebx, 0

jl exits

mov eax, [edi+ebx\*4]

mov [ebp-8], eax #local varuable2

addint:

mov eax, [ebp-4]

cmp eax, [ebp-8] #сравниваем единичный интервал с текущей левой границей

jl checklgrint #если левее, то берем следующую левую границу

mov ecx, [esi+edx\*4] #кол-во чисел на единичном интервале

mov eax, [ebp+28]

add [eax+ebx\*4], ecx #к результату на промежутке прибавляем кол-во на каждом единичном интервале

dec dword ptr [ebp-4] #переход к следующему числу

dec edx

jmp addint

exits:

pop edi

pop esi

pop edx

pop ecx

pop ebx

pop eax

add esp, 8

pop ebp

ret

**Тестирование**

Введите количество чисел NumRanDat (>0, <=16000)

15000

Введите интервал случайных чисел[Xmin, Xmax]

-5000

7000

Введите количество промежутков разбиения NInt (>0, <=24)

3

Введите левую границу LGrInt[1], должна принадлежать [-5000,7000]

-2000

Введите левую границу LGrInt[2], должна принадлежать [-5000,7000]

1000

Введите левую границу LGrInt[3], должна принадлежать [-5000,7000]

4000

4

Интервал0 -5000 3794

Интервал1 -2000 3762

Интервал2 1000 3642

Интервал3 4000 3802

Введите количество чисел NumRanDat (>0, <=16000)

10000

Введите интервал случайных чисел[Xmin, Xmax]

-1000

12874

Введите количество промежутков разбиения NInt (>0, <=24)

5

Введите левую границу LGrInt[1], должна принадлежать [-1000,12874]

-1000

Введите левую границу LGrInt[2], должна принадлежать [-1000,12874]

0

Введите левую границу LGrInt[3], должна принадлежать [-1000,12874]

178

Введите левую границу LGrInt[4], должна принадлежать [-1000,12874]

9879

Введите левую границу LGrInt[5], должна принадлежать [-1000,12874]

12874

4

Интервал0 -1000 0

Интервал1 -1000 724

Интервал 2 0 126

Интервал3 178 6948

Интервал4 9879 2202

Интервал5 12874 0

**Вывод**

Программа написана на высокого уровня С++ и использует 2 ассемблерные процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы.

Результатом работы программы является текстовая таблица, содержащая номер интервала, левую границу, количество псевдослучайных чисел, попавших в этот интервал. Таблица выводится на экран и сохраняется в файл.