**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 целых чисел в заданные интервалы**

|  |  |  |
| --- | --- | --- |
| Студент гр. 7383 |  | Кирсанов А. Я. |
| Преподаватель |  | Кирьянчиков В. А. |

Санкт-Петербург

2017

**Цель работы.**

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

**Формулировка задания.**

На языке высокого уровня программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение.

Далее должны вызываться две 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.

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.

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.

**Тестирование.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер массива  псевдосл.  чисел | Диапазон  [Xmin,Xmax] | Массив распределения по интервалам единичной длины | Массив левых границ | Результирующий массив частотного распределения |
| 10 | [0;10] | (0) 0  (1) 1  (2) 1  (3) 1  (4) 1  (5) 1  (6) 1  (7) 1  (8) 1  (9) 2  (10) 0 | 1) 0  2) 3  3) 5  4) 7  5) 9 | [0,2] 2  [3,4] 2  [5,6] 2  [7,8] 2  [9,10] 2 |
| 100 | [-5;5] | (-5) 0  (-4) 1  (-3) 1  (-2) 1  (-1) 1  (0) 1  (1) 1  (2) 1  (3) 1  (4) 2  (5) 0 | 1) -5  2) 0  3) 5 | [-5,-1] 43  [0,3] 33  [4,5] 24 |
| 15 | [-5;-3] | (-5) 5  (-4) 1  (-3) 9 | 1) -5 | [-5,-3] 15 |

**Выводы.**

В ходе выполнения данной лабораторной работы была написана программа на языке Ассемблера, которая строит частотное распределение попаданий псевдослучайных чисел в заданные интервалы.

В результате выполнения лабораторной работы были получены практические навыки программирования на языке Ассемблер.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД**

**lr6.cpp**

#include "pch.h"

#include <iostream>

#include <random>

#include <fstream>

#include <ctime>

using namespace std;

#define NUMBER 16000 //максимальное длина числа

#define BORD 24 //максимальное количество интервалов

void get\_arr(int & amount, int\* & array, int & Xmin, int & Xmax, int & Border, int\* & LeftBorder);

void generation(int\* & array, int amount, int min, int max);

void res\_func1(int Xmin, int\* array, int amount);

void res\_func2(int Xmax, int\* LeftBorder, int\* array, int amount);

ofstream fout("res.txt");

extern "C"

{

void func1(int array[], int amount, int counter[], int Xmin);

void func2(int With1Range[], int LeftBorder[], int InterDif[], int Border, int Xmin);

}

int main(void) {

setlocale(LC\_ALL, "rus");

int amount = 0; //кол-во псевдослучайных чисел

int Xmin = 0, Xmax = 0, MainRange = 0;

int Border = 0; //кол - во границ

int \*array = NULL; //массив чисел

int \*LeftBorder = NULL; //массив левых границ

int \*RightBorder = NULL; //массив правых границ

int \*counter = NULL; //для интервалов единичной длины

int\* InterDif = NULL; //для интервалов заданной длины

/\*Получение псевдослучайных чисел\*/

get\_arr(amount, array, Xmin, Xmax, Border, LeftBorder);

/\*Создание массива правых границ, исходя из

массива левых границ и граничного значения Xmax\*/

RightBorder = new int[Border];

for (int i = 0; i < Border - 1; i++)

RightBorder[i] = LeftBorder[i + 1] - 1;

RightBorder[Border - 1] = Xmax;

MainRange = Xmax - Xmin + 1;

counter = new int[MainRange] {0};

InterDif = new int[Border] {0};

func1(array, amount, counter, Xmin); //распределение по ед. интервалам

func2(counter, RightBorder, InterDif, Border, Xmin); //распределение по опр. интервалам

res\_func1(Xmin, counter, MainRange); //вывод результата первой процедуры

res\_func2(Xmax, LeftBorder, InterDif, Border); //вывод результата второй процедуры

system("pause");

return 0;

}

void get\_arr(int & amount, int \*&ArrNumber, int &Xmin, int &Xmax, int &Border, int \*&LeftBorder)

{

do {

cout << "Введите количество случайных чисел, 0 < N <= " << NUMBER << ": ";

cin >> amount;

if (amount <= 0 || amount > NUMBER)

cout << "\nОшибка диапазона!\n\n";

} while (amount <= 0 || amount > NUMBER);

ArrNumber = new int[amount];

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, amount, Xmin, Xmax);

do {

cout << "\nВведите количество интервалов разбиения заданного диапазона ( 0 < N <= " << BORD << "): ";

cin >> Border; cout << endl;

if (Border <= 0 || Border > BORD)

cout << "\nОшибка: количество интервалов не входит в указанный диапазон!Повторите попытку.\n";

} while (Border <= 0 || Border > BORD);

LeftBorder = new int[Border];

cout << "\nВвод интервалов по возрастанию (1-ый интервал равен левой границе)\n";

LeftBorder[0] = Xmin; //левая граница

cout << "Граница 1: " << Xmin << "\n";

int tmp = 0;

for (int i = 1; i < Border; i++)

{

do {

cout << "Граница " << i + 1 << ": ";

cin >> tmp;

if (tmp <= LeftBorder[i - 1] || tmp >= Xmax)

{

cout << "\n\nВыход за пределы диапазона!\n\n";

}

else

{

LeftBorder[i] = tmp;

break;

}

} while (true);

}

}

void generation(int\* & array, int len, int min, int max)

{

random\_device rd; // класс, который 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++) {

array[i] = distr(gen);

}

}

void res\_func1(int Xmin, int\* counter, int amount)

{

cout << "\n\*\*\*Распределение случайных чисел по интервалам единичной длины\*\*\*\n"; fout << "\n\*\*\*Распределение случайных чисел по интервалам единичной длины\*\*\*\n";

cout << "Число\t|Кл-во\n"; fout << "№\tЧисло\tКл-во\n";

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

cout << Xmin + i << "\t|" << counter[i] << '\n'; fout << i + 1 << '\t' << Xmin + i << "\t" << counter[i] << '\n';

}

}

void res\_func2(int Xmax, int\* LeftBorder, int\* InterDif, int amount)

{

cout << "\n\*\*\*Распределение случайных чисел по заданным интервалам\*\*\*\n"; fout << "\n\*\*\*Распределение случайных чисел по заданным интервалам\*\*\*\n";

cout << "№\t|[левая;правая гр]\t|Количество\t\n"; fout << "№\t|[левая;правая гр]\t|Количество\t\n";

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

{

cout << i << "\t" << LeftBorder[i] << "\t\t"; fout << i << "\t" << LeftBorder[i] << "\t\t";

if (i == amount - 1)

{

cout << Xmax; fout << Xmax;

}

else

{

cout << LeftBorder[i + 1] - 1; fout << LeftBorder[i + 1] - 1;

}

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

}

}

**6.asm**

.386p

.MODEL FLAT;, C

.CODE

PUBLIC func1 ; C func1

func1 PROC C array:DWORD, amount:DWORD, counter:DWORD, Xmin:DWORD

MOV EDI, array ;Адрес массива случайных чисел

MOV ESI, counter ;Адрес массива счетчика чисел

MOV ECX, amount ;Длина массива случайных чисел

MOV EAX, Xmin

CYCLE:

MOV EBX, [EDI] ;Извлечение случайного числа N

SUB EBX, EAX ;Вычесть левую границу 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 func2

func2 PROC C counter:DWORD, RightBorder:DWORD, InterDif:DWORD, Border:DWORD, Xmin:DWORD

MOV EDI, RightBorder ;Адрес массива правых границ

MOV ESI, counter ;Адрес массива счетчика чисел

MOV EAX, InterDif ;Адрес массива заданных интервалов

MOV ECX, Border ;Количество разбиений (интервалов)

MOV EBX, XMIN

XOR EDX, EDX

CYCLE:

CMP EBX, [EDI]

JG NEXT\_RANGE ;Переход, если число больше текущ. границы

ADD EDX, [ESI] ;Накопление

INC EBX ;Переход к следующему числу

ADD ESI, 4 ;Переход к след. эл. распр. чисел с ед. диапазоном

JMP CYCLE

NEXT\_RANGE: ;Достигнута правая граница интервала

MOV [EAX], EDX ;Помещаем в массив с зад. 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