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.

Студент гр. 1383 _____ Куликов М.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Реализовать программу построения частотного распределения попаданий псевдослучайных чисел в заданные интервалы на ЯВУ и языке Ассемблера.

Задание.

Вариант 2.

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.

Выполнение работы.

Программа состоит из трех модулей, два которых написаны на языке Ассемблера, а один на C++.

В модуле на C++ происходит вывод информации пользователю, считывание входных данных и их проверка на корректность. В этом модуле также вызываются процедуры asm_func1 и asm_func2, которые обрабатывают входные данные.

В процедуре asm_func1 формируется распределение исходных чисел по интервалам единичной длины в заданной пользователем области. Результат разбиения записывается в result1.

В процедуре asm_func2 формируется окончательное распределение по заданным пользователем интервалам с помощью левых границ этих интервалов и ранее полученного массива result1 с единичным распределением. Количества чисел на единичных интервалах прибавляются к значениям, ответственным за количество чисел на пользовательских интервалах основываясь на числах, за которые эти единичные интервалы отвечают. Результат работы процедуры записывается в массив result2.

Выводы.

Была реализована программа построения частотного распределения попаданий псевдослучайных чисел в заданные интервалы на ЯВУ и языке Ассемблера.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab6.cpp

```
#include <iostream>
#include <random>
#include <fstream>
#include <windows.h>
#include <time.h>


extern "C" void asm_func1(int* arr, int arr_size, int* result1, int Xmin);
extern "C" void asm_func2(int single_arrange,int tmp, int* result1, int* result2, int* LGrInt, int Nint, int Xmin);

int cmp(const void* a, const void* b)
{
   return (*(int*)a - *(int*)b);
}

int main() {
   SetConsoleOutputCP(1251);
   SetConsoleCP(1251);
   int arr_size, Nint, Xmin,Xmax;
   int* arr;
   int* LGrInt;
   int* result1;
   int* result2;

   std::cout << "Введите размер длины массива псевдослучайных чисел: \n";
```

```
std::cin >> arr_size;
if (arr_size <= 0 || arr_size > 16 * 1024) {
   printf("Вы ввели неверную длину массива.\n");
   return 0;
}
arr = new int[arr_size];

std::cout << "Введите левую границу массива\n";
std::cin >> Xmin;

std::cout << "Введите правую границу массива\n";
std::cin >> Xmax;

if (Xmin > Xmax) {
   printf("Введенный диапазон чисел некорректен\n");
   return 0;
}

std::cout << "Введите количество интервалов разбиения\n";
std::cin >> Nint;

if (Nint <= 0) {
   printf("Введенное количество интервалов разбиения некорректное\n");
   return 0;
}

LGrInt = new int[Nint];
std::cout << "Введите левые границы интервалов разбиения \n";
for (int i = 0; i < Nint; i++) {
   std::cin >> LGrInt[i];
```

```
    if (LGrInt[i] < Xmin || LGrInt[i] > Xmax) {
        printf("Введенныт границы некорректны\n");
        return 0;
    }
}

qsort(LGrInt, Nint, sizeof(int), cmp);


result1 = new int[abs(Xmax - Xmin) + 1];
result2 = new int[Nint];
int single_arrange = abs(Xmax - Xmin) + 1;
for (int i = 0; i < single_arrange; i++) {
    result1[i] = 0;
}
for (int i = 0; i < Nint; i++) {
    result2[i] = 0;
}

std::cout << "Случайно сгенерированные числа: \n";
srand(time(NULL));
for (int i = 0; i < arr_size; i++) {
    arr[i] = Xmin + rand() % (Xmax - Xmin + 1);
    std::cout << arr[i] << " ";
}

asm_func1(arr, arr_size, result1, Xmin);
```

```
        std::cout << "\nПромежуточное распределение с единичным интервалом:
\n";
        for (int i = 0; i < single_arrange; i++) {
            std::cout << result1[i] <<" ";
        }

        int tmp = Xmin;
        asm_func2(single_arrange,tmp,result1, result2, LGrInt, Nint, Xmin);
        std::cout << "\n";

        std::cout << "Частотное распределение чисел по интервалам: \n";

        for (int i = 0; i < Nint; i++) {
            std::cout << i << " " << LGrInt[i] << " " << result2[i] << "\n";
        }
        std::ofstream file("output.txt");
        for (int i = 0; i < Nint; i++) {
            file << i << " " << LGrInt[i] << " " << result2[i] << "\n";
        }
        file.close();
        }
```

Название файла: asm_func1.asm

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

PUBLIC C asm_func1
asm_func1 PROC C array: dword, arr_size: dword, result1: dword, Xmin: dword

push eax
push ebx
push ecx
push esi
push edi

mov esi, array
mov edi, result1
mov ecx, arr_size

spread_for_one:
        mov eax, [esi]
        sub eax, Xmin
        mov ebx, [edi+4*eax]
        inc ebx
        mov [edi+4*eax], ebx
        add esi, 4
        loop spread_for_one
```

```
pop edi
pop esi
pop ecx
pop ebx
pop eax

ret
asm_func1 endp
end
```

Название файла: asm_func2.asm

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

PUBLIC C asm_func2
asm_func2 PROC C single_arrange: dword, tmp: dword,result1: dword, result2: dword, LGrInt: dword, Nint: dword, Xmin: dword

push eax
push ebx
push ecx
push esi
push edi

mov esi, LGrInt
mov edi, result2
mov ecx, Nint
```

```
cmp_loop:
        mov eax,[esi]
        mov ebx,[esi+4]

        push esi
        mov esi,result1
        compare:
                cmp tmp,eax
                jl lower_or_addited
                cmp ecx,1
                je addition
                cmp tmp,ebx
                je next_cycle
        addition:
                push eax
                mov eax,tmp
                sub eax,Xmin
                push ebx
                mov ebx,[esi+4*eax]
                add [edi],ebx
                pop ebx
                pop eax
                cmp ecx,1
                jne lower_or_addited
                dec single_arrange
                inc tmp
                cmp single_arrange,0
                jne addition
```

```
            jmp next_cycle


        lower_or_addited:
            dec single_arrange
            inc tmp
            jmp compare




        next_cycle:
            pop esi
            add esi,4
            add edi,4
            loop cmp_loop



pop edi
pop esi
pop ecx
pop ebx
pop eax

ret
asm_func2 endp
```

```
end
```