

**МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ**

**ОТЧЕТ
по лабораторной работе №4
по дисциплине «Организация ЭВМ и системы»
Тема: Представление и обработка символьной информации с
использованием строковых команд**

Студента гр. 1383

Самулевич С.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

ФОРМУЛИРОВКА ЗАДАНИЯ

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 ($\leq 16K$, $K=1024$) 2. Диапазон изменения массива псевдослучайных целых чисел $[Xmin, Xmax]$, значения могут быть биполярные; 14 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

В файле *lab6.cpp* реализовано считывание исходных данных на ЯВУ, также происходит выделение памяти под массив вывода *res* и генерация массива *arr* псевдослучайных целых чисел, генерирующихся при помощи стандартной функции *std::uniform_int_distribution*. Далее происходит вызов процедуры *inter* частотного распределения чисел по интервалам, реализованной на языке ассемблера при помощи *extern*. В файле *modul.asm* происходит проверка каждого числа на принадлежность к одному из интервалов при помощи команды *loop*,

(аналога цикла *for*) и сравнения каждого числа с каждой из границ. В случае, если элемент оказывается меньше, то происходит выход из цикла, так как верхняя граница найдена; Если же верхнюю границу найти не удалось, то элемент записывается в последний интервал. После чего происходит запись чисел в соответствующие ячейки массива *res*.

Программный код см. в приложении А

Выводы

В ходе работы были изучены способы совместной разработки на ЯВУ и языке ассембле

Приложение А

Исходный код программы

Название файла lab6.cpp

```
#include <iostream>
#include <fstream>
#include <random>

extern "C" void inter(int* arr, int len, int* LGrInt, int* res);

int main() {
    int NumRanDat;

    int NInt;
    int Xmin;
    int Xmax;

    puts("Total amount");
    std::cin >> NumRanDat;
    int* arr = new int[NumRanDat];
    puts("Min number");
    std::cin >> Xmin;
    puts("Max number");
    std::cin >> Xmax;
    puts("Number of intervals");
    std::cin >> NInt;
    puts("Left borders");
    int* LGrInt = new int[NInt];
    for (int i = 0; i < NInt; i++) {
        std::cin >> LGrInt[i];
    }
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> distrib(Xmin, Xmax);

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

```

        arr[i] = distrib(gen);
    }

    for (int i = 0; i < NumRanDat; i++) {
        printf("%d ", arr[i]);
    }
    puts("");

    int* res = new int[NInt];
    for (int i = 0; i < NInt; i++) {
        res[i] = 0;
    }

    inter(arr, NumRanDat, LGrInt, res);

    for (int i = 0; i < NInt; i++) {
        printf("%d ", res[i]);
    }

}

```

Название файла modul.asm

.586p

.model flat, c

.code

```

inter proc c uses edi esi, arr:dword, len:dword, LGrInt:dword,
res:dword

```

push eax

push ebx

push ecx

push edi

push esi

mov esi, arr

mov edi, LGrInt

mov ecx, len

```
sub eax, eax

for0:
sub ebx, ebx
mov eax, len
sub eax, ecx

for1:

push ecx
mov ecx, [esi + 4*eax]
cmp ecx, [edi + 4*ebx]
pop ecx

jl end_for1
inc ebx
jmp for1

end_for1:

mov edi, res
mov eax, [edi + 4*ebx]
inc eax
mov [edi + 4*ebx], eax
mov edi, LGrInt

loop for0

pop esi
pop edi
pop ecx
pop ebx
pop eax

ret

inter endp
end
```