

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 построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы

Студент гр. 0383

Тарасов К.О.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы

Разработать программу построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы, связав модуль на Ассемблере с файлом на ЯВУ

Задание

На языке С программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND_GEN (при 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 целых чисел - NumRanDat ($\leq 16K$)
2. Диапазон изменения массива псевдослучайных целых чисел $[X_{min}, X_{max}]$ (м.б. биполярный, например, $[-100, 100]$)
3. Массив псевдослучайных целых чисел $\{X_i\}$.
4. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (≤ 24)
5. Массив левых границ интервалов разбиения LGrInt .

В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину, левые границы могут задаваться в произвольном порядке и иметь произвольные значения. Если $X_{\min} < LGrInt(1)$, то часть данных не будет участвовать в формировании распределения. Каждый интервал, кроме последнего, следует интерпретировать как $[LGrInt(i), LGrInt(i+1))$. Если у последнего интервала правая граница меньше X_{\max} , то часть данных не будет участвовать в формировании распределения.

Результаты:

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.

Вариант работы — 7

Вид распределения — нормальный, число процедур — 2, $N_{int} \geq D_x$, $Lg1 > X_{\min}$, $ПГ_{\text{посл}} \leq X_{\max}$

Ход работы

Программа была реализована из 3-х модулей, и на C++ и 2 на ассемблере. На ЯВУ написана программа, которая перенаправляет информацию введенную пользователем с консоли и перенаправляет её в ассемблерные модули, а так же выводит результаты работы программы в консоль и в файл.

Первый ассемблерный модуль реализует распределение чисел по единичным отрезкам. Циклически записывается в новый массив количество повторений каждого числа. Второй модуль формирует распределение тех же чисел, но по уже заданным интервалам.

Связь между модулями осуществлена с помощью спецификатора `extern`, который позволяет выполнять раздельную компиляцию модулей

Тестирование

Табл. 1. Результат тестирования.

Вызванные команды	Результат	Комментарий																																							
NumDatRan=10 xmin = 0 xmax = 10 Nint = 10 LgrInt = (1 2 3 4 5 6 7 8 9 10)	<table> <tr> <th>№</th><th>Граница</th><th>Количество чисел</th></tr> <tr><td>1</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>2</td><td>3</td></tr> <tr><td>3</td><td>3</td><td>0</td></tr> <tr><td>4</td><td>4</td><td>1</td></tr> <tr><td>5</td><td>5</td><td>0</td></tr> <tr><td>6</td><td>6</td><td>2</td></tr> <tr><td>7</td><td>7</td><td>1</td></tr> <tr><td>8</td><td>8</td><td>1</td></tr> <tr><td>9</td><td>9</td><td>0</td></tr> <tr><td>10</td><td>10</td><td>1</td></tr> </table>	№	Граница	Количество чисел	1	1	0	2	2	3	3	3	0	4	4	1	5	5	0	6	6	2	7	7	1	8	8	1	9	9	0	10	10	1	Верно						
№	Граница	Количество чисел																																							
1	1	0																																							
2	2	3																																							
3	3	0																																							
4	4	1																																							
5	5	0																																							
6	6	2																																							
7	7	1																																							
8	8	1																																							
9	9	0																																							
10	10	1																																							
NumDatRan = 10 xmin = 0 xmax = 10 Nint = 9	Число интервалов должно быть больше или равно Dx(Xmax-Xmin)	Верно																																							
NumDatRan = 12 xmin = -6 xmax = 6 Nint = 12 LgrInt = (-5 -4 -3 -2 -1 0 1 2 3 4 5 6)	<table> <tr> <th>№</th><th>Граница</th><th>Количество чисел</th></tr> <tr><td>1</td><td>-5</td><td>1</td></tr> <tr><td>2</td><td>-4</td><td>1</td></tr> <tr><td>3</td><td>-3</td><td>0</td></tr> <tr><td>4</td><td>-2</td><td>2</td></tr> <tr><td>5</td><td>-1</td><td>2</td></tr> <tr><td>6</td><td>0</td><td>0</td></tr> <tr><td>7</td><td>1</td><td>1</td></tr> <tr><td>8</td><td>2</td><td>2</td></tr> <tr><td>9</td><td>3</td><td>0</td></tr> <tr><td>10</td><td>4</td><td>2</td></tr> <tr><td>11</td><td>5</td><td>0</td></tr> <tr><td>12</td><td>6</td><td>1</td></tr> </table>	№	Граница	Количество чисел	1	-5	1	2	-4	1	3	-3	0	4	-2	2	5	-1	2	6	0	0	7	1	1	8	2	2	9	3	0	10	4	2	11	5	0	12	6	1	Верно
№	Граница	Количество чисел																																							
1	-5	1																																							
2	-4	1																																							
3	-3	0																																							
4	-2	2																																							
5	-1	2																																							
6	0	0																																							
7	1	1																																							
8	2	2																																							
9	3	0																																							
10	4	2																																							
11	5	0																																							
12	6	1																																							

Выводы:

В ходе выполнения работы была реализована программа частотного распределения случайных чисел по заданным интервалам на языке C++ с использованием ассемблерных модулей

ПРИЛОЖЕНИЕ А

Текст компонентов программы lr5

main.cpp

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <random>
```

```
#include <time.h>
```

```
using namespace std;
```

```
extern "C" void first(int* numbers, int numbers_size, int* result, int xmin);
```

```
extern "C" void second(int* array, int array_size, int xmin, int* intervals, int intervals_size, int* result);
```

```
int main() {
```

```
    setlocale(0, "Russian");
```

```
    srand(time(NULL));
```

```
    ofstream result("result.txt");
```

```
    int numbers_size;
```

```
    int* numbers;
```

```
    int xmin, xmax;
```

```
    int intervals_size;
```

```
    int* intervals;
```

```
    int* intervals2;
```

```
    int* mod1_result;
```

```
    int* mod2_result;
```

```
    cout << "Введите количество чисел:\n";
```

```
    cin >> numbers_size;
```

```

if (numbers_size > 16 * 1024) {
    cout << "Количество чисел должно быть меньше или равно, чем 16*1024\n";
    return 0;
}
cout << "Введите xmin и xmax:\n";
cin >> xmin >> xmax;
int Dx = xmax - xmin;
if (Dx > 24) {
    cout << "Xmax и Xmin не должны отличаться больше чем на 24\n";
    return 0;
}

cout << "Введите число границ:\n";
cin >> intervals_size;

if (intervals_size > 24) {
    cout << "Число интервалов должно быть меньше или равно 24\n";
    return 0;
}
if (intervals_size < Dx) {
    cout << "Число интервалов должно быть больше или равно Dx(Xmax-Xmin)\n";
    return 0;
}

numbers = new int[numbers_size];
intervals = new int[intervals_size];
intervals2 = new int[intervals_size];

int len_asm_mod1_res = abs(xmax - xmin) + 1;
mod1_result = new int[len_asm_mod1_res];
for (int i = 0; i < len_asm_mod1_res; i++)
    mod1_result[i] = 0;

mod2_result = new int[intervals_size + 1];
for (int i = 0; i < intervals_size + 1; i++)

```

```
mod2_result[i] = 0;
```

```
cout << "Введите все границы:\n";  
for (int i = 0; i < intervals_size; i++) {  
    cin >> intervals[i];  
    if (intervals[i] <= xmin) {  
        cout << "Левая граница должна быть больше Xmin\n";  
        return 0;  
    }  
    intervals2[i] = intervals[i];  
}
```

```
std::random_device rd;  
std::mt19937 gen(rd());  
std::uniform_int_distribution<> dis(xmin, xmax);  
for (int i = 0; i < numbers_size; i++) numbers[i] = dis(gen);
```

```
cout << "Сгенерированные значения\n";  
result << "Сгенерированные значения\n";  
for (int i = 0; i < numbers_size; i++) {  
    cout << numbers[i] << ' ';  
    result << numbers[i] << ' ';  
}  
cout << '\n';  
cout << '\n';  
result << '\n';  
result << '\n';
```

```
first(numbers, numbers_size, mod1_result, xmin);
```

```
second(mod1_result, numbers_size, xmin, intervals, intervals_size, mod2_result);
```

```

cout << "Результат:\n";
result << "Результат:\n";
cout << "№\tГраница\tКоличество чисел" << endl;
result << "№\tГраница\tКоличество чисел" << endl;

for (int i = 1; i < intervals_size + 1; i++) {
    if (i != intervals_size) {
        cout << i << "\t" << intervals2[i - 1] << '\t' << mod2_result[i] << endl;
        result << i << "\t" << intervals2[i - 1] << '\t' << mod2_result[i] << endl;
    }
    else {
        cout << i << "\t" << xmax << '\t' << mod2_result[i] << endl;
        result << i << "\t" << xmax << '\t' << mod2_result[i] << endl;
    }
}

delete[] numbers;
delete[] intervals;
delete[] intervals2;
delete[] mod1_result;
delete[] mod2_result;

return 0;
}

```

fisrt.asm

.586p

.MODEL FLAT, C

.CODE

PUBLIC C first

first PROC C array: dword, arraysize: dword, res: dword, xmin: dword

push esi

push edi


```

mov edi, array
mov ecx, arraysize
mov esi, res

for_numbers:
    mov eax, [edi]
    sub eax, xmin
    mov ebx, [esi + 4*eax]
    inc ebx
    mov [esi + 4*eax], ebx
    add edi, 4
    loop for_numbers

```

```

pop edi
pop esi

```

```

ret
first ENDP
END

```

second.asm

.586p

```

.MODEL FLAT, C
.CODE
PUBLIC C second

```

```

second PROC C array: dword, array_size: dword, xmin: dword, borders: dword, intN: dword,
result: dword

```

```

push esi
push edi
push ebp

```

```

mov edi, array
mov esi, borders
mov ecx, intN

```

```

for_borders:
    mov eax, [esi]
    sub eax, xmin
    mov [esi], eax
    add esi, 4

```

```

        loop for_borders

mov esi, borders
mov ecx, intN
mov ebx, 0
mov eax, [esi]

for_loop:
    push ecx
    mov ecx, eax
    push esi
    mov esi, result

    for_array:
        cmp ecx, 0
        je end_for
        mov eax, [edi]
        add [esi + 4*ebx], eax
        add edi, 4
        loop for_array

    end_for:
        pop esi
        inc ebx
        mov eax, [esi]
        add esi, 4
        sub eax, [esi]
        neg eax
        pop ecx
        loop for_loop

mov esi, result
mov ecx, intN
mov eax, 0

fin_for:
    add eax, [esi]
    add esi, 4
    loop fin_for

mov esi, result
sub eax, array_size
neg eax

add [esi + 4*ebx], eax

pop ebp
pop edi
pop esi

ret
second ENDP

```

END