

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.**

Студентка гр. 0383

Преподаватель

Сергеев Д.В.

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Вариант 14: 1 + - + - - +

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.

Выполнение работы.

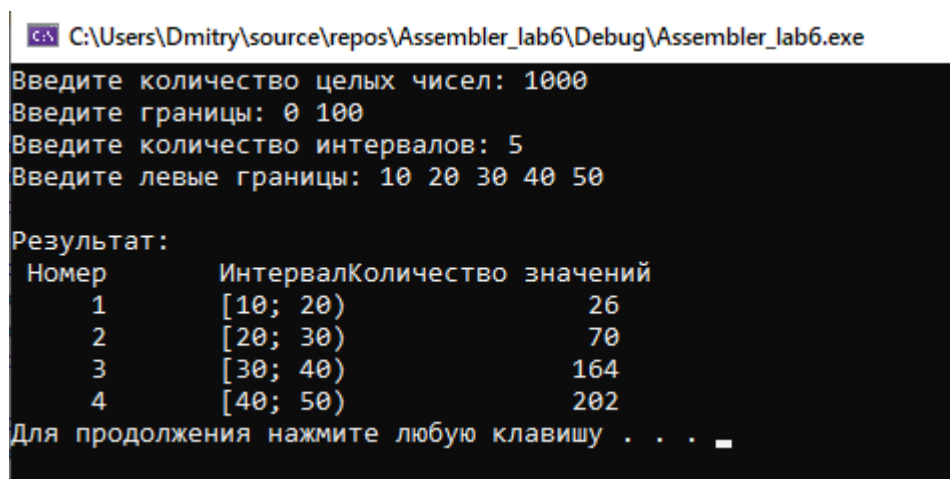
В начале программы на языке C++ происходит считывание входных данных, таких как кол-во генерируемых чисел, границы распределения, кол-во интервалов и интервалы. Установлена поддержка русского языка и обращение к пользователю. Каждому интервалу присваивается свой индекс, для дальнейшей обработки. Также присутствует проверка на выполнение требования $N_{int} \geq X_{max} - X_{min}$.

Также в программе рассчитываются математическое ожидание и среднеквадратичное отклонения для нормального распределения, далее происходит генерация псевдослучайных чисел. После вызывается ассемблерный модуль, подсчитывающий количество вхождений в каждый интервал посредством сравнений. Для каждого элемента по очереди происходит поиск интервала, в который он входит, а количество вхождений для этого интервала увеличивается на единицу.

После работы ассемблерного модуля результат его работы выводится в виде таблицы на экран и записывается в файл.

Тексты исходного файла программы см. в приложении А.

Рис. 1 - Проверка работы программы.



```
C:\Users\Dmitry\source\repos\Assembler_lab6\Debug\Assembler_lab6.exe
Введите количество целых чисел: 1000
Введите границы: 0 100
Введите количество интервалов: 5
Введите левые границы: 10 20 30 40 50

Результат:
  Номер    Интервал    Количество значений
    1      [10; 20)         26
    2      [20; 30)         70
    3      [30; 40)        164
    4      [40; 50)        202
Для продолжения нажмите любую клавишу . . .
```

Рис. 2 - Проверка работы программы.

```
C:\Users\Dmitry\source\repos\Assembler_lab6\Debug\Assembler_lab6.exe
Введите количество целых чисел: 10000
Введите границы: 0 100
Введите количество интервалов: 5
Введите левые границы: 0 20 40 60 80

Результат:


| Номер | Интервал | Количество значений |
|-------|----------|---------------------|
| 1     | [0; 20)  | 306                 |
| 2     | [20; 40) | 2310                |
| 3     | [40; 60) | 4508                |
| 4     | [60; 80) | 2453                |


Для продолжения нажмите любую клавишу . . .
```

Выводы.

В ходе выполнения данной лабораторной работы был изучен принцип организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А

ТЕКСТЫ ИСХОДНЫХ ФАЙЛОВ ПРОГРАММЫ

Название файла: lab6.cpp

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

using namespace std;

extern "C" void func(int* nums, int numsCount, int* leftBorders, int* result);

void output(string A, string B, string C, ofstream& file) {
    cout << setw(6) << right << A << setw(15) << right << B << setw(17) << right << C <<
endl;
    file << setw(6) << right << A << setw(15) << right << B << setw(17) << right << C << endl;
}

int main() {
    setlocale(LC_ALL, "ru");

    int randNumCount;
    int max_x, min_x;
    int intervalCount;
    cout << "Введите количество целых чисел: ";
    cin >> randNumCount;
    while (randNumCount <= 0) {
        cout << "Некорректное количество чисел, попробуйте еще раз.\nВведите количество
целых чисел: ";
```

```
    cin >> randNumCount;
};
```

```
cout << "Введите границы: ";
cin >> min_x >> max_x;
while (max_x <= min_x) {
    cout << "Некорректное количество границ, попробуйте еще раз.\nВведите границы: ";
    cin >> min_x >> max_x;
};
```

```
cout << "Введите количество интервалов: ";
cin >> intervalCount;
while (randNumCount <= 0 or !(max_x - min_x > intervalCount)) {
    cout << "Некорректное количество интервалов, попробуйте еще раз.\nВведите
количество интервалов: ";
    cin >> intervalCount;
};
```

```
cout << "Введите левые границы: ";
int* leftBorders = new int[intervalCount];
int* result = new int[intervalCount];
for (int i = 0; i < intervalCount; i++) {
    cin >> leftBorders[i];

    int index = i;
    while (index && leftBorders[index] < leftBorders[index - 1]) {
        swap(leftBorders[index--], leftBorders[index]);
    }
    result[i] = 0;
```

```

}
cout << endl;

random_device rd{};
mt19937 gen(rd());

float expectation = float(max_x + min_x) / 2; // мат ожидание
float stddev = float(max_x - min_x) / 6; // мат отклонение
normal_distribution<float> dist(expectation, stddev);

int* nums = new int[randNumCount];
for (int i = 0; i < randNumCount; i++) {
    nums[i] = round(dist(gen));
}

func(nums, randNumCount, leftBorders, result);

ofstream file("output.txt");
cout << "Результат:\n";
output("Номер", "Интервал", "Количество значений", file);
for (int i = 0; i < intervalCount - 1; i++) {
    output(
        to_string(i + 1),
        '[' + to_string(leftBorders[i]) + "; " + to_string(leftBorders[i + 1]) + ")",
        to_string(result[i + 1]),
        file
    );
}

```



```
file.close();  
system("pause");  
return 0;  
}
```

Название файла: sort.asm

.586

.MODEL FLAT, C

.CODE

func PROC C nums:dword, numsCount:dword, leftBorders:dword, result:dword

push eax

push ebx

push ecx

push edx

push esi

push edi

mov ecx, numsCount

mov esi, nums

mov edi, leftBorders

mov edx, 0

next:

mov ebx, [esi+4*edx]

cmp ebx, [edi]

j1 continue

mov eax, 0

searchInterval:

 cmp ebx, [edi+4*eax]

 j1 endSearch

 inc eax

 jmp searchInterval

endSearch:

mov edi, result

mov ebx, [edi+4*eax]

inc ebx

mov [edi+4*eax], ebx

mov edi, leftBorders

continue:

inc edx

loop next

pop edi

pop esi

pop edx

pop ecx

pop ebx

pop eax

ret

func ENDP

END