

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 распределения попаданий
псевдослучайных целых чисел в заданные интервалы
Вариант 20

Студент гр. 0382

Шангичев В. А.

Преподаватель

Ефремов М. А.

Санкт-Петербург

2018

Цель работы.

Изучение основных принципов организации связи ассемблера с ЯВУ.
Написание программы построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

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
[Xmin, Xmax] (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.

Задание на разработку программы выбирается из таблицы 1 в зависимости от номера студента в группе. Варианты заданий различаются:

- 1) видом распределения псевдослучайных чисел: равномерное или нормальное (гаусовское);
- 2) количеством ассемблерных модулей, формирующих требуемое распределение:
 - если указан 1 модуль, то он сразу формирует распределение по заданным интервалам и возвращает его в главную программу, написанную на ЯВУ;
 - если указаны 2 модуля, то первый из них формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат (это распределение должно выводиться на экран для контроля); затем вызывается второй модуль который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами).Это распределение возвращается в главную программу и выдается как основной результат в виде текстового файла.
- 3) условием – может ли число интервалов быть больше-равно ($N_{int} \geq D_x$) или меньше ($N_{int} < D_x$) диапазона изменения входных чисел;
- 4) условием – может ли первая левая граница быть больше X_{min} ($Lg1 > X_{min}$) или могут ли какие-то левые границы быть меньше X_{min} ($Lgi \leq X_{min}$);

- 5) условием – может ли правая граница последнего интервала быть больше X_{\max} ($ПГ_{\text{посл}} > X_{\max}$) или меньше-равна X_{\max} ($ПГ_{\text{посл}} \leq X_{\max}$).

Вариант 20

Вид распределения: нормальное.

Число ассемблерных процедур: 2.

$N_{\text{int}} < D_x$ и $L_{gi} \leq X_{\min}$.

Выполнение работы.

В ходе выполнения лабораторной работы было написано три файла: `main.cpp`, `units_counting.asm`, `intervals_counting.asm`.

Рассмотрим файл `main.cpp`. После подключения всех необходимых заголовочных файлов объявляются две внешние функции: `unit_distribution` и `intervals_distribution`. Эти функции описаны в файлах `units_counting.asm` и `intervals_counting.asm` соответственно. В главной функции программы происходит считывание количества генерируемых чисел, их минимальное и максимальное значения, интервалы. После этого происходит генерация случайных чисел в соответствии с заданными ограничениями. После этого происходит вызов функции `unit_distribution`. После вызова данной функции массив `units` будет содержать в каждой ячейке частоту появления числа в сгенерированной последовательности (со смещением в `x_min`). После этого вызывается функция `intervals_distribution`, и выводится ответ.

Рассмотрим файл `units_counting.asm`, содержащий реализацию функции `unit_distribution`. Данная функция принимает на вход массив сгенерированных чисел, их количество, массив для записи результата и ограничения этих чисел снизу. В теле функции происходит перебор всех чисел

в массиве с записью каждого из них в соответствующую ячейку в массиве результата (`[сгенерированное число] – [x_min]`).

Рассмотрим файл `intervals_counting`, содержащий описание функции `intervals_distribution`, принимающей на вход массив интервалов, их количество, массив `units`, его размер, минимальное значение и массив для записи результата. В данной функции посредством массива `units` подсчитываются количество всех чисел в заданном интервале и записывается в массив для записи результата.

Тестирование.

Тестирование см. в приложении Б. По результатам тестирования был сделан вывод, что программа работает верно.

Выводы.

В ходе работы были изучены основные принципы организации связи ассемблера с ЯВУ. Была написана программа, частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.cpp

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

using namespace std;

extern "C" void unit_distribution(int* numbers, int n, int* res, int
x_min, float* mid);
extern "C" void intervals_distribution(int* intervals, int n_int, int*
units, int n_units, int x_min, int* res);

int main() {
    int n, x_min, x_max, n_int;
    cout << "Enter amount of numbers:" << endl;
    cin >> n;
    cout << "Enter Xmin and Xmax seperated by space:" << endl;
    cin >> x_min >> x_max;
    cout << "Enter number of intervals:" << endl;
    cin >> n_int;
    if (n_int >= (x_max - x_min)) {
        cout << "Nint < D_x must be" << endl;
        return 0;
    }
    cout << "Enter Lgi seperated by spaces:" << endl;
    auto intervals = new int[n_int + 1];
    for (int i = 0; i < n_int; ++i) {
        cin >> intervals[i];
    }
    intervals[n_int] = x_max;

    double mean = (x_min + x_max) / 2;
    double sigma = (x_max - x_min) / 4;
    if (!sigma) sigma = 1;
    cout << "\nNumbers generated with normal distribution (mean = "
<< mean << ", sigma = " << sigma << ").\n" << endl;
    random_device r_d;
    mt19937 generator(r_d());
    normal_distribution<double> distribution(mean, sigma);

    auto numbers = new int[n];
    for (int i = 0; i < n; ++i) {
        int number = distribution(generator);
        while (number < x_min || number > x_max)
            number = distribution(generator);
        numbers[i] = number;
        cout << number << " ";
        if ((i + 1) % 50 == 0) cout << "\n";
    }
```

```

    }
    cout << endl;

    auto units = new int[x_max - x_min + 1];
    auto result = new int[n_int + 1];
    for (int i = 0; i < x_max - x_min + 1; ++i)
        units[i] = 0;
    for (int i = 0; i < n_int + 1; ++i)
        result[i] = 0;
    float mid = 0.0;
    unit_distribution(numbers, n, units, x_min);

    cout << "units:" << "\n";
    for (int i = 0; i < x_max - x_min + 1; i++) {
        cout << units[i] << " ";
    }

    intervals_distribution(intervals, n_int, units, x_max - x_min +
1, x_min, result);
    cout << "\n\n";

    auto head = "N\tLeft border\tAmount of numbers";
    cout << head << endl;
    for (int i = 0; i < n_int; i++) {
        auto row = to_string(i + 1) + "\t" + to_string(intervals[i])
+ "\t\t" + to_string(result[i]) + "\n";
        cout << row;
    }

    return 0;
}

```

Файл units_counting.asm

.MODEL FLAT, C

.DATA

.CODE

PUBLIC C unit_distribution

unit_distribution PROC C numbers: dword, n: dword, res: dword, xmin: dword

mov esi, numbers

mov edi, res

mov ecx, n

mov edx, 0h

start_loop:

mov eax, [esi]

sub eax, xmin

mov ebx, [edi + 4*eax]

add ebx, 1

mov [edi + 4*eax], ebx

add esi, 4

loop start_loop

```
ret
unit_distribution ENDP
END
```

Файл intervals_counting.asm

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

```
PUBLIC C intervals_distribution
intervals_distribution PROC C intervals: dword, n_int: dword, units: dword, n_units:
dword, x_min: dword, res: dword
    mov esi, intervals
    mov edi, res
    mov ecx, n_int
```

```
start_loop:
    mov eax, [esi]
    add esi, 4h
    mov ebx, [esi]
    sub ebx, eax
    sub eax, x_min
```

```
    cmp eax, 0
    jge to_units
    neg eax
    sub ebx, eax
    mov eax, 0h
    cmp ebx, 0
    jg to_units
    sub ecx, 1
    add edi, 4h
    jmp start_loop
```

```
to_units:
    push ecx
    push esi

    mov ecx, ebx
    mov ebx, 0h
    mov esi, units
```

```
start_loop2:
    add ebx, [esi + eax*4]
    add eax, 1
    loop start_loop2
    mov [edi], ebx
    add edi, 4h
```

```
pop esi
pop ecx
```

```
loop start_loop
```

```
sub edi, 4h
mov esi, units
```

```
mov ebx, [esi + eax*4]
add [edi], ebx
```

```
ret
```



```
intervals_distribution ENDP  
END
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Тест 1.

```
Enter amount of numbers:
30
Enter Xmin and Xmax seperated by space:
5 15
Enter number of intervals:
5
Enter Lgi seperated by spaces:
-7 -3 0 6 14

Numbers generated with normal distribution (mean = 10, sigma = 2).
9 14 9 10 9 12 5 7 13 8 6 10 9 6 8 10 9 9 12 10 11 13 11 9 11 9 9 7 11 12

N      Left border      Amount of numbers
1      -7                0
2      -3                0
3       0                1
4       6               28
5      14                1
```

Тест 2.

```
Enter amount of numbers:
25
Enter Xmin and Xmax seperated by space:
50 100
Enter number of intervals:
5
Enter Lgi seperated by spaces:
1 10 30 40 60

Numbers generated with normal distribution (mean = 75, sigma = 12).
97 74 74 71 85 69 79 77 88 65 77 72 79 69 86 85 66 80 61 65 72 82 72 51 78

N      Left border      Amount of numbers
1       1                0
2      10                0
3      30                0
4      40                1
5      60               24
```

Тест 3.

```
15
Enter Xmin and Xmax seperated by space:
0 100
Enter number of intervals:
10
Enter Lgi seperated by spaces:
0 10 20 30 40 50 60 70 80 90

Numbers generated with normal distribution (mean = 50, sigma = 25).
44 53 0 17 76 78 41 47 76 40 32 56 39 74 1

N      Left border      Amount of numbers
1       0                2
2      10                1
3      20                0
4      30                2
5      40                4
6      50                2
7      60                0
8      70                4
9      80                0
10     90                0
```