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

Студент гр. 0382 _____ Самулевич В.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Научится писать и компилировать многофайловые программы, имеющие два типа файлов: .cpp(на языке C++) и .s(на языке ассемблера).

Задание.

Исходные данные:

1. Длина массива псевдослучайных целых чисел - NumRanDat (<= 16K)
2. Диапазон изменения массива псевдослучайных целых чисел
   [Xmin, Xmax] (м.б. биполярный, например, [-100, 100])
3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24)
4. 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 Xmin < 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 Xmax, 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 16:

- Для генерации случайных чисел использовать нормальное(Гауссово) распределение.

- В программе должно быть два ассемблерных модуля: 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 выводится 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).

- Число интервалов меньше диапазона изменения входных чисел.

- Левые границы интервалов могут быть меньше левой границы диапазона изменения псевдослучайных чисел(Xmin).

- Правая граница последнего интервала меньше или равна правой границе диапазона изменения псевдослучайных чисел(Xmax).

Выполнение работы.

Программа состоит из 3 файлов: main.cpp, func1.s и func2.s. Рассмотрим за что отвечает каждый из них:

func1.s содержит определение функции void func1(int* input, int len, int* result, int min), где input-массив случайных чисел,len-его длина, result-массив для записи результата(распределение исходных чисел по интервалам единичной длины или, что равносильно, сколько раз каждое число из диапазона [Xmin, Xmax] встречается в input), min – левая граница диапазона, в котором генерировались случайные числа. Общий принцип работы func1 в следующем: с помощью цикла обходятся все элементы в input, и для каждого из них

вычисляется адрес ячейки в result, которая соответствует текущему значению (для этого из значения вычитается min и результат умножается на 4),после чего число по полученному адресу инкрементируется.

func2.s описывает функцию void func2 (int* res_1, int* intervals, int* res_2, int right_border, int len, int min), где res_1 – результат работы func1, intervals - массив левых границ интервалов, res_2 – массив для записи результата, right_border – правая граница последнего интервала, len – количество интервалов, min - левая граница диапазона, в котором генерировались случайные числа. Алгоритм работы: В цикле для каждого элемента x_i из intervals (кроме последнего) считаем сумму ячеек от res_1[$x_i + min$] до res_1[$x_{i+1} + min$] (не включая res_1[$x_{i+1} + min$]),после чего записываем результат в res_2[i].Если оказывается, что какие-либо номера ячеек получились меньше 0, берем номер равным 0. После цикла проделываем аналогичные действия для последнего элемента intervals, только вместо x_{i+1} берем значение right_border и при вычислении суммы учитываем значение res_1[right_border + min] .

В main.cpp происходит считывание входных данных, проверка их на корректность, генерация массива псевдослучайных чисел с помощью класса std::normal_distribution, вызов ассемблерных модулей и вывод результатов на экран, а также запись их в файл.

Исходный код программы см. в приложении А.

Тестирование.

Результаты тестирования программы представлены в приложении В.

Выводы.

Были изучены способы написания и сборки многофайловых программ, состоящих из .cpp и .s файлов. Помимо этого была разработана программа,

строящая частотное распределение попаданий псевдослучайных чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <random>
using namespace std;

extern "C" void func1(int* input,int len, int* result,int min);
extern "C" void func2(int* res_1, int* intervals, int* res_2,int
right_border,int len,int min);

int cmp(const void* x1,const void* x2){
    return ( *(int*) x1 - *(int*) x2 );

}


int main(){

    cout<<"Number of random numbers: ";
    int len;
    while(true){
        cin >> len;
        if(len > 0)
            break;

        cout<<"Incorrect quantity, enter a new value: ";
    }

   cout<<"Enter the change range: ";
    int min;
    int max;
    while(true){
        cin >> min;
        cin >> max;
        if(min <= max)
            break;
        cout<<"Invalid range, enter new values: ";
    }

    cout<<"Enter the number of intervals: ";
    int intervals_count;
    while(true){
        cin >> intervals_count;
        if(intervals_count > 0)
            break;
        cout<<"invalid quantity, enter a new value: ";
    }
    int intervals[intervals_count];
    int res_2[intervals_count];
    cout<<"Enter the left borders of the intervals: ";
    for(int i=0;i<intervals_count;i++){
```

```
        cin >> intervals[i];
        res_2[i]=0;
    }



    qsort((void*)intervals,intervals_count,sizeof(int),cmp);
    int right_border;
    cout<<"Enter the right border of the last interval: ";
    while(true){
        cin >> right_border;
        if(right_border >= intervals[intervals_count-1] && right_border
<=max+1)
            break;
        cout<<"invalid size, enter a new value: ";
    }


    int* numbers=new int[len];
    random_device rd;
    mt19937 gen(rd());
    float mean=float(max+min)/2;
    float stddev=float(max-min)/6;
    normal_distribution<float> dist(min, max);
        int j=0;
        while(j < len){
            int new_val=round(dist(gen));
            if(new_val <= max && new_val >= min){
                numbers[j] = new_val;
                j++;
            }
        }


    if(len < 101){
        cout<<"generated values:\n";
        for(int i=0;i<len;i++)
            cout<<numbers[i]<<", ";
        cout<<'\n';
    }


    cout<<right_border<<'\n';
    int* res_1=new int[max-min+1];
    for(int i=0;i<max-min+1;i++)
        res_1[i]=0;
    func1(numbers,len,res_1,min);
    cout<<right_border<<'\n';
    cout<<"Distribution over intervals of length 1: \n";
    for(int i=0;i<max-min+1;i++){
        cout<<i+min<<':'<<res_1[i]<<' ';
    }

    cout<<"\n\n";

    func2(res_1,intervals,res_2,right_border,intervals_count,min);
    ofstream file;
```

```
    file.open("result.txt");
    string title= "N\tLeft border\tcount\t\n";
    cout<<title;
    file<<title;
    for(int i=0;i<intervals_count;i++){
        string res_string;
        res_string=res_string+to_string(i)+'\t'+to_string(intervals[i]);
        res_string=res_string+"\t\t"+to_string(res_2[i])+'\n';
        cout<<res_string;
        file<<res_string;
    }
    file.close();
}
```

Название файла: func1.s

```
intel_syntax noprefix
    .text
    .globl func1
    .type func1,@function

func1:
    push rbp
    mov  rbp,rsp
    mov QWORD PTR -8[rbp], rdi
    mov DWORD PTR -12[rbp], esi
    mov QWORD PTR -24[rbp], rdx
    mov DWORD PTR -16[rbp], ecx

    mov ecx,-12[rbp] #len
    mov rsi, -8[rbp] #input
    mov rdi, -24[rbp] #result
    mov rax,0
start:
    mov eax,[rsi]
    sub eax,-16[rbp]
    imul eax,4
    inc DWORD PTR [rdi+rax]
    add rsi,4
    loop start
    pop rbp
ret
```

Название файла: func2.s

```
.intel_syntax noprefix
    .text
    .globl func2
    .type func2,@function

func2:
    push rbp
    mov rbp,rsp

    mov QWORD PTR -8[rbp], rdi #res_1
    mov QWORD PTR -16[rbp], rsi #intervals
    mov QWORD PTR -24[rbp], rdx #res_2
```

```
    mov DWORD PTR -28[rbp], ecx #right_border
    mov DWORD PTR -32[rbp], r8d #len
    mov DWORD PTR -36[rbp], r9d #min

    mov rdi,-24[rbp] #res_2
    mov rsi,-8[rbp] #res_1
    sub rsi,4
    mov rbx, -16[rbp] #intervals
    mov ecx, -32[rbp] #intervals len
    dec ecx
    mov edx, -36[rbp] #min
    dec edx


skip:
    inc edx
    add rsi,4
    cmp DWORD PTR [rsi],0
    je skip


    cmp edx,DWORD PTR [rbx]
    jge next
    mov edx,[rbx]
next:

    cmp ecx,0
    je end


start:
    add rbx,4
    mov eax,[rbx]

sum:
    cmp edx,eax
    jge continue
    mov r8d,[rsi]
    add [rdi],r8d
    inc edx
    add rsi,4
    jmp sum

continue:
    add rdi,4
    loop start


end:
    mov eax,-28[rbp]
last_sum:
    cmp edx,eax
    jg return
    mov r8d,[rsi]
    add [rdi],r8d
    inc edx
    add rsi,4
    jmp last_sum
```

```
return:
pop rbp
ret
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

```
Number of random numbers: 1000
Enter the change range: -10 10
Enter the number of intervals: 2
Enter the left borders of the intervals: -75 -9
Enter the right border of the last interval: 8
8
8
Distribution over intervals of length 1:
-10:69 -9:70 -8:78 -7:67 -6:76 -5:80 -4:70 -3:63 -2:77 -1:55 0:45 1:49 2:39 3:35 4:31 5:24 6:20 7:17 8:13 9:13 10:9

N       Left border     count
0       -75             69
1       -9              909
```

Рисунок 1.Результаты работы программы на наборе входных данных №1.

```
Number of random numbers: 2000
Enter the change range: 0 100
Enter the number of intervals: 10
Enter the left borders of the intervals: -10 0 10 20 30 40 50 60 70 80
Enter the right border of the last interval: 90
90
90
Distribution over intervals of length 1:
0:38 1:23 2:23 3:16 4:17 5:22 6:20 7:25 8:28 9:20 10:13 11:23 12:23 13:13 14:30 15:25 16:25 17:31 18:29 19:17 20:24 21:21 22:26 23:23 24:24 25
:21 26:25 27:22 28:28 29:23 30:16 31:23 32:22 33:24 34:12 35:20 36:20 37:19 38:20 39:20 40:18 41:30 42:32 43:22 44:19 45:21 46:20 47:29 48:21
49:16 50:18 51:18 52:18 53:14 54:25 55:28 56:27 57:19 58:14 59:24 60:20 61:17 62:24 63:12 64:21 65:18 66:15 67:9 68:15 69:20 70:23 71:22 72:11
 73:13 74:21 75:14 76:20 77:20 78:19 79:18 80:14 81:15 82:10 83:23 84:15 85:15 86:12 87:14 88:21 89:18 90:14 91:12 92:17 93:13 94:12 95:10 96:
16 97:22 98:22 99:14 100:12

N       Left border     count
0       -10             0
1       0               232
2       10              229
3       20              237
4       30              196
5       40              228
6       50              205
7       60              171
8       70              181
9       80              171
```

Рисунок 2.Результаты работы программы на наборе входных данных №2.

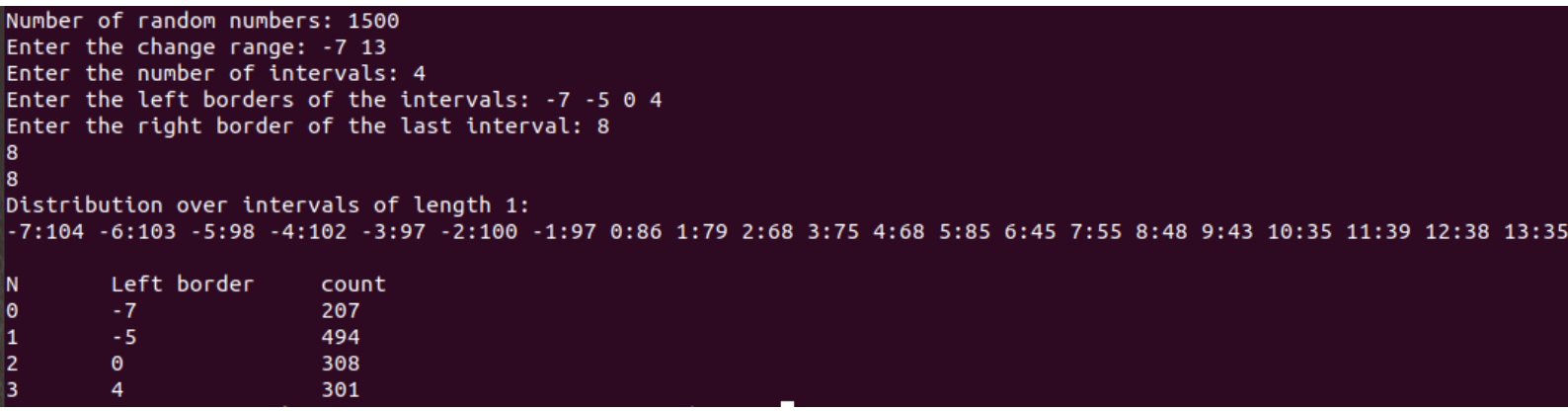

Рисунок 3.Результаты работы программы на наборе входных данных №3.