

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 програм-
мы построения частотного распределение попаданий псевдослучай-
ных целых чисел в заданные интервалы.

Студент гр. 0383

Зенин П.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Научиться создавать программы с использованием комбинации языков (C++ и Ассемблер).

Задание.

На языке C 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.

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} ($\Pi\Gamma_{\text{посл}} > X_{\max}$) или меньше-равна X_{\max} ($\Pi\Gamma_{\text{посл}} \leq X_{\max}$).

Таблица 1. $D_x = X_{\max} - X_{\min}$; Lg1, Lgi –первая или любая левая граница; ПГ-
 посл – правая граница последнего интервала

Вид	Чис-	Nint	Nint	Lgi ≤	Lg1 >	ПГ-	
распреде-	ло	ассем.	≥ Dx	< Dx	Xmin	Xmin	посл
рав-	1		+	-	-	+	-
нор-	1		+	-	-	+	-
рав-	2		-	+	+	-	+
нор-	2		-	+	+	-	+
рав-	1		-	+	+	-	-
нор-	1		-	+	+	-	-
рав-	2		+	-	-	+	+
нор-	2		+	-	-	+	+
рав-	1		-	+	-	+	-
нор-	1		-	+	-	+	-
рав-	2		-	+	+	-	+
нор-	2		-	+	+	-	+
рав-	1		+	-	+	-	-
нор-	1		+	-	+	-	-
рав-	2		+	-	+	-	+
нор-	2		+	-	+	-	+
рав-	1		-	+	-	+	-
нор-	1		-	+	-	+	-
рав-	2		-	+	+	-	+
нор-	2		-	+	+	-	-
рав-	1		+	-	-	+	-
нор-	1		+	-	-	+	+
рав-	2		+	-	+	-	-
нор-	2		+	-	+	-	-
рав-	1		-	+	-	+	-
нор-	1		-	+	-	+	-
рав-	2		-	+	+	-	+
нор-	2		+	-	+	-	-
рав-	1		+	-	+	-	-
нор-	1		+	-	+	-	-
рав-	2		-	+	+	-	+
нор-	2		+	-	+	-	-
рав-	1		+	-	+	-	-

Замечания:

1) На ЯВУ следует реализовать только ввод исходных данных (возможно с контролем), вывод и генерацию псевдослучайных целых чисел. Всю остальную функциональность следует программировать на ассемблере.

2) В отладочной версии программы (при небольшом количестве псевдослучайных чисел, не превышающем 100 значений) для контроля работы датчика сгенерированные числа, приведенные к целому виду, следует выводить на экран или в файл. В основной версии программы, предоставляемой для защиты, вывод сгенерированных псевдослучайных чисел выполнять не нужно.

Вариант 3:

Распределение — равномерное

Число процедур: 2

$N_{int} < D_x$,

$L_{gi} \leq X_{min}$,

$ПГ_{посл} \leq X_{max}$.

Выполнение работы.

Для работы данной программы пользователю требуется ввести количество псевдослучайных чисел, границы их распределения, количество интервалов разбиения и левые границы интервалов. Данные переменные вводятся пользователем с клавиатуры, ввод осуществляется при помощи функции языка C++ - `cin`. После ввода производится проверка допустимости входных данных. Если данные недопустимы, то выводится сообщение в терминал. После окончания ввода производится генерация N случайных чисел при помощи функции языка C++. Эти значения заносятся в массив. Далее производится обработка полученных значений в модулях, написанных на языке Ассемблера.

Первый модуль выводит частоты встречаемости каждого целого числа в диапазоне, лежащем в границах интервала генерации чисел. Подсчёт производится следующим образом. Создаётся массив, где каждому элементу интервала соответствует число, обозначающее количество раз, когда число встретилось. В процедуре происходит проход по массиву сгенерированных чисел. Затем, при встрече конкретного числа, выполняется инкрементация соответствующей ячейки в массиве результатов.

Второй модуль находит частоты вхождения сгенерированных чисел в конкретные интервалы. Делается это следующим образом: На языке C++ создаётся массив результатов, в каждой ячейки которой хранится количество элементов, которые попали в интервал, затем берётся число из массива сгенерированных чисел. После происходит проверка принадлежности того или иного числа к данному интервалу. Если число лежит меньше правой границы интервала (или равен правой

границе интервала, если интервал последний), то происходит инкрементация соответствующей ячейки массива результатов.

Код программы см. в приложении А.

Выводы.

В этой работе была изучена работа с модулями, написанными на языке Ассемблера.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ И ДРУГИЕ ФАЙЛЫ

Название файла: lab6.cpp

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

using namespace std;

extern "C" void func1(int* X, int n, int* res1, int x_min);
extern "C" void func2(int* res1, int* GrInt, int* res2, int x_max, int x_min, int n);

int main() {

    setlocale(LC_ALL, "ru");

    int NumRamDat;
    cout << "Введите количество псевдослучайных чисел:\n";
    cin >> NumRamDat;
    if (NumRamDat <= 0 || NumRamDat > 16000) {
        cout << "Недопустимое значение количества псевдослучайных чисел (0;16000]\n";
        return 1;
    }

    int Xmax, Xmin;
    cout << "Введите границы генерации псевдослучайных чисел в формате <Xmin Xmax>\n";
    cin >> Xmin >> Xmax;
    if (Xmax <= Xmin) {
        cout << "Недопустимые границы\n";
        return 1;
    }

    int NInt;
    cout << "Введите количество интервалов разбиения:\n";
    cin >> NInt;
    if (NInt <= 0 || NInt > 24) {
        cout << "Количество интервалов должно быть от 1 до 24 ";
        return 1;
    }
}
```



```

    if (NInt > abs(Xmax - Xmin)+1) {
        cout << "Количество интервалов не может быть больше диапазона генерации псевдо-
случайных чисел\n";
        return 1;
    }

    int* LGrInt = new int[NInt + 1];
    cout << "Введите левые границы:\n";
    for (int i = 0; i < NInt; i++) {
        cin >> LGrInt[i];

        if (LGrInt[i] > Xmax) {
            cout << "Недопустимое значение интервала\n";
            return 1;
        }

        int ind = i;
        while (ind && LGrInt[ind] < LGrInt[ind - 1]) {
            swap(LGrInt[ind--], LGrInt[ind]);
        }
    }
    LGrInt[NInt] = Xmax+1;
    //if (LGrInt[0] < Xmin)
    //  LGrInt[0] = Xmin;

    int* X = new int[NumRamDat];
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> distrib(Xmin, Xmax);

    cout << "Массив псевдослучайных чисел:\n";
    for (int i = 0; i < NumRamDat; i++) {
        X[i] = distrib(gen);
        cout << X[i] << ' ';
    }
    cout << '\n';

    int* res_1 = new int[Xmax - Xmin + 1];
    for (int i = 0; i < (Xmax - Xmin + 1); i++)
        res_1[i] = 0;

    int* res_2 = new int[NInt+1];

```

```

    for (int i = 0; i < NInt; i++)
        res_2[i] = 0;

    func1(X, NumRamDat, res_1, Xmin);
    cout << "Частоты встречаемости чисел:\n";
    for (int i = 0; i < (Xmax - Xmin + 1); i++)
        cout << i + Xmin << ": " << res_1[i] << "; \n";
    cout << "\n";

    func2(res_1, LGrInt, res_2, Xmax, Xmin, NInt);
    ofstream out;
    out.open("C:\\Users\\Peter\\Desktop\\out.txt");
    cout << "Распределение чисел по интервалам:\n";
    out << "Распределение чисел по интервалам:\n";
    cout << "Номер интервала Левая граница Количество чисел\n";
    out << "Номер интервала Левая граница Количество чисел\n";
    for (int i = 0; i < NInt; i++) {
        cout << setw(8) << i << setw(14) << LGrInt[i] << setw(16) << res_2[i] << "\n";
        out << setw(8) << i << setw(14) << LGrInt[i] << setw(16) << res_2[i] << "\n";
    }
    out.close();

    return 0;
}

```

Файл: func1.asm

```

.586

.MODEL FLAT, C

.CODE

PUBLIC C func1

func1 PROC C X:dword, n: dword, res1: dword, x_min: dword

    push esi
    push edi

    mov esi, X
    mov edi, res1
    mov ecx, n

change:

```

```

mov eax, [esi]
sub eax, x_min
mov ebx, [edi + 4*eax]
inc ebx
mov [edi + 4*eax], ebx
add esi, 4
loop change

```

```
pop edi
```

```
pop esi
```

```
ret
```

```
func1 ENDP
```

```
END
```

Файл: func2.asm

```
.586
```

```
.MODEL FLAT, C
```

```
.CODE
```

```
PUBLIC C func2
```

```
func2 PROC C res1:dword, GrInt: dword, res2: dword, x_max: dword, x_min: dword, n: dword
```

```
push esi
```

```
push edi
```

```
mov esi, GrInt
```

```
mov edi, res2
```

```
mov ecx, n
```

```
lp:
```

```
mov eax, [esi] ; левая граница интервала
```

```
mov ebx, [esi + 4] ; правая граница
```

```
cmp eax, x_min ; если eax >= x_min
```

```
jge l2
```

```
mov eax, 0 ; иначе, eax = 0, начало массива res1
```

```
sub ebx, x_min ; если длина интервала = 0
```

```
jle l4
```

```
jmp l5
```

```
l2:
```

```
sub ebx, eax ; количество элементов в интервале
```

```
    cmp ebx, 0 ; если длина интервала = 0
    je l4
    sub eax, x_min ; индекс первого элемента из текущего интервала в массиве res1
```

15:

```
    push esi
    push ecx
```

```
    mov ecx, ebx ; количество элементов из res1 по которым нужно пройти
    mov esi, res1 ; массив
    mov ebx, 0 ; считает сумму подходящих элементов
```

lp2: ; цикл, считает сумму элементов, входящих в интервал

```
    add ebx, [esi + 4*eax]
    inc eax
    loop lp2
```

pop ecx

cmp ecx, 0 ; если обрабатывали не последний элемент, то записываем сумму в массив
результат

jne l3

add ebx, [esi + 4*eax] ; иначе, скобка последнего интервала вадратная, поэтому до-
бавляем еще элемент

l3:

```
    mov [edi], ebx ; записываем результат
    pop esi
    jmp l1
```

l4:

mov [edi], ebx ; записываем 0, если интервал пустой

l1:

```
    add edi, 4 ; двигаемся к след. элементам массивов
    add esi, 4
```

loop lp

pop edi

pop esi

ret

func2 ENDP

END