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.

Студент гр. 9382 _____ Павлов Р.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Написать программу на ЯВУ и подключить к ней ассемблерные модули с функциями обработки последовательности псевдослучайных чисел.

Задание (вариант 2).

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND_GEN (при его отсутствии программу датчика получить у преподавателя). Далее должен вызываться ассемблерный модуль(модули) для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину. 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.

Ход работы.

1) На ЯВУ написана программа, отвечающая за генерацию последовательности псевдослучайных чисел в заданном диапазоне и распределение его на интервалы.

2) Создан ассемблерный модуль с двумя процедурами:

1. Процедура подсчёта попадания чисел в интервалы единичной длины в данном диапазоне.
2. Процедура подсчёта попадания чисел в интервалы произвольной длины, заданные пользователем или равномерно распределённые по диапазону.

3) Организована связь программы на ЯВУ с модулем посредством вызова процедур из основной программы.

Выводы.

В результате выполнения лабораторной работы написана программа, создающая последовательность случайных чисел и обрабатывающая эту последовательность с помощью реализованных ассемблерных модулей.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД

- имя файла : ASM2.cpp

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

//Объявление ассемблерных процедур
extern "C" {
	void single(int* main_arr, int main_len, int* single_counter_arr, int min);
	void custom(int* single_counter_arr, int single_counter_len, int*
left_borders_arr, int* custom_counter_arr, int custom_counter_len, int min);
}

//Заполнение массива случайными числами
void GetRandomNum(int*& Number, int len, int min, int max){
	std::random_device rd;
	std::mt19937 gen(rd());
	std::uniform_int_distribution<> distr(min, max);

	for (--len;len >= 0;len--) {
		Number[len] = distr(gen);
	}
}

//Вставка значения (границы) в массив
bool InsertBorder(int*& LGrInt, int cur_len, int num, int min, int max) {
	if (num < min || num > max) {
		std::cout << "Ошибка: число не в пределах интервала или уже задано\n";
		return false;
	}
	int i = 0;
	int j = 0;
	while (i < cur_len) {
		if (LGrInt[i] < num) {
			i++;
		}
		else if (LGrInt[i] == num) {
			std::cout << "Ошибка: число уже присутствует в массиве\n";
			return false;
		}
		else {
			j = cur_len;
			while (i < j) {
				LGrInt[j] = LGrInt[j - 1];
				j--;
			}
			break;
		}
	}

	LGrInt[i] = num;
	return true;
}

//Получение от пользователя информации о числах и интервалах
bool GetInformation(int& NumRanDat, int*& Number, int& Xmin, int& Xmax, int& NInt, int*&
LGrInt){
```

```
	int len = 0;
	int i = 0;
	setlocale(LC_ALL, "rus");

	std::cout << "Введите количество случайных чисел (от 1 до 16384): ";
	std::cin >> NumRanDat;
	std::cout << NumRanDat << std::endl;

	while (!(NumRanDat > 0 && NumRanDat < 16385)){
		std::cout << "Число не входит в диапазон\n" << "Повторите ввод: ";
		std::cin >> NumRanDat;
	}

	std::cout << "Введите диапазон случайных чисел: \n" << "От: ";
	std::cin >> Xmin;
	std::cout << "До :";
	std::cin >> Xmax;

	while (Xmax <= Xmin){
		std::cout << "Правая граница диапазона должна быть больше левой\n" << "По-
вторите ввод: ";
		std::cin >> Xmax;
	}

	Number = new int[NumRanDat];

	if (!Number) {
		std::cout << "Не удалось выделить память для массива чисел\n";
		return false;
	}

	GetRandomNum(Number, NumRanDat, Xmin, Xmax);

	std::cout << "Введите количестно интервалов для разделения диапазона (от 1 до 24):
";
	std::cin >> NInt;

	while (NInt <= 0 || NInt > 24){
		std::cin.clear();
		std::cin.sync();
		std::cout << "Количество интервалов не входит в указанный диапазон\n" <<
"Повторите ввод: ";
		std::cin >> NInt;
	}
	LGrInt = new int[NInt];

	if (!LGrInt) {
		std::cout << "Не удалось выделить память для массива левых границ\n";
		return 0;
	}

	std::cout << "Выбор интервалов.\n" << "\t1.Распределить интервалы равномерно по
диапазону\n" << "\t2.Установить интервалы самостоятельно\n";

	while (i == 0){
		std::cin >> i;

		switch (i){
		case 1:
			len = Xmax - Xmin;
			for (i = 0;i < NInt;i++) {
				LGrInt[i] = Xmin + len / NInt * i;
```

```
                }

                break;
        case 2:
                LGrInt[0] = Xmin;
                std::cout << "Граница 1: " << Xmin << "\n\n";
                for (i = 1;i < NInt;i++) {
                        do {
                                std::cout << "Граница " << i + 1 << ": ";
                                std::cin >> len;
                        } while (!InsertBorder(LGrInt, i, len, Xmin, Xmax));
                }

                break;
        default:
                std::cout << "Недопустимый номер операции, повторите ввод: ";
                i = 0;

                break;
        }
    }
    return true;
}

//Инициализация нулями
void InitArray(int*& Arr, int len){
    for (--len;len >= 0;len--) {
        Arr[len] = 0;
    }
}

//Вывод результата для единичных интервалов
void PrintResult1(int* Number, int len, int min){
    int i = 0;
    std::ofstream fout;
    fout.open("result1.txt");

    if (!fout.is_open()) {
        std::cout << "Не удалось открыть файл.\n";
        return;
    }

    std::cout << "Распределение случайных чисел по интервалам единичной длины:\n";
    std::cout << "№\tЛевая гр.\tКол-во\t\n";
    fout << "Распределение случайных чисел по интервалам единичной длины:\n";
    fout << "№\tЛевая гр.\tКол-во\t\n";

    for (i = 0;i < len;i++)
    {
        std::cout << i << '\t' << min + i << "\t\t" << Number[i] << '\n';
        fout << i << '\t' << min + i << "\t\t" << Number[i] << '\n';
    }

    std::cout << "-----------------------------\n";
    fout.close();
}

//Вывод результатов для различных интервалов
void PrintResult2(int* LGrInt, int* CountNum, int len){
    int i = 0;
    std::ofstream fout;
    fout.open("result2.txt");
```

```
	if (!fout.is_open()) {
		std::cout << "Не удалось открыть файл.\n";
		return;
	}

	std::cout << "Распределение случайных чисел по заданным интервалам:\n";
	std::cout << "№\tЛевая гр.\tКол-во\t\n";
	fout << "Распределение случайных чисел по интервалам единичной длины:\n";
	fout << "№\tЛевая гр.\tКол-во\t\n";

	for (i = 0;i < len;i++) {
		std::cout << i << '\t' << LGrInt[i] << "\t\t" << CountNum[i] << '\n';
		fout << i << '\t' << LGrInt[i] << "\t\t" << CountNum[i] << '\n';
	}

	fout.close();
}

//Вывод массива на экран
void PrintArray(int* array, int length) {
	for (int i = 0; i < length; i++) {
		std::cout << i << ") " << array[i] << "\n";
	}
}

int main(void) {
	int NumRanDat = 0;
	int* Number = nullptr;
	int Xmin = 0;
	int Xmax = 0;
	int NInt = 0;
	int* LGrInt = nullptr;
	int* CountNumUnit1 = nullptr;
	int lenUnit1 = 0;
	int* CountNumN = nullptr;

	//Ввод информации о массиве

	if (!GetInformation(NumRanDat, Number, Xmin, Xmax, NInt, LGrInt)) {
		return 1;
	}
	PrintArray(Number, NumRanDat);

	//Создание необходимых массивов

	lenUnit1 = Xmax - Xmin + 1;
	CountNumUnit1 = new int[lenUnit1];
	if (!CountNumUnit1) {
		std::cout << "Ошибка: не удалось выделить память\n";
		return 1;
	}
	InitArray(CountNumUnit1, lenUnit1);

	CountNumN = new int[NInt];
	if (!CountNumN) {
		std::cout << "Ошибка: не удалось выделить память\n";
		return 1;
	}
	InitArray(CountNumN, NInt);

	//Распределение и подсчёт
```

```
	single(Number, NumRanDat, CountNumUnit1, Xmin);
	custom(CountNumUnit1, lenUnit1, LGrInt, CountNumN, NInt, Xmin);

	//Вывод на экран и в файл

	PrintResult1(CountNumUnit1, lenUnit1, Xmin);
	PrintResult2(LGrInt, CountNumN, NInt);

	delete[] CountNumN;
	delete[] CountNumUnit1;
	delete[] Number;
	delete[] LGrInt;

	return 0;
}
```

Имя файла: distr.asm

```
.586p
.model flat, c

.data
	SUPER_counter dd 0

.code
	public c single ;распределение по интервалам единичной длины
	single proc c main_arr:dword, main_len:byte, single_counter_arr:dword,
min:byte
		push edi
		push esi
		push eax
		push ebx
		push ecx
		push edx

		mov eax, dword ptr min		; сохраняем адреса начала массивов
чисел и счётчика,
		mov ecx, dword ptr main_len	; а также минимальное значение и
длину массива чисел
		mov edi, main_arr
		mov esi, single_counter_arr

		counter:
			mov ebx,[edi]				; получаем текущее чис-
ло
			sub ebx,eax				; получаем смещение от-
носительно начала счётчика ( ЗНАЧЕНИЕ - XMIN )
			mov edx,[esi+4*ebx]			; берём значение для
этого числа, находящееся в счётчике
			inc edx					; отмечаем, что
встретилось [ещё] один раз
			mov [esi+4*ebx],edx			; записываем значение в
массив-счётчик
			add edi,4				; идём к следующему
элементу
			loop counter

		pop edx
		pop ecx
```

```
            pop ebx
            pop eax
            pop esi
            pop edi

            ret
      single endp

      public c custom ;распределение по различным интервалам
      custom   proc   c   single_counter_arr:dword,   single_counter_len:byte,
left_borders_arr:dword,   custom_counter_arr:dword   ,custom_counter_len:byte,
min:byte
            push edi
            push esi
            push eax
            push ebx
            push ecx
            push edx

            mov ecx,dword ptr single_counter_len            ;   счётчик   для
цикла

            mov edi,dword ptr single_counter_len            ;   указатель   на
последний элемент массива-счётчика вхождений чисел (интервалы единичной дли-
ны)
            dec edi
            shl edi,2
            add edi,single_counter_arr

            mov eax, dword ptr custom_counter_len           ;  смещение  отно-
сительно начала двух массивов : левых границ и счётчика попаданий в интервал
            dec eax
            shl eax,2

            push edi
            mov esi, left_borders_arr
            mov edi, custom_counter_arr
            add edi,eax
            add esi,eax                                 ; запись в ESI указате-
ля на последний элемент массива левых границ
            mov SUPER_counter,edi               ; и запись в память указателя
на последний элемент счётчика попаданий
            pop edi

            sub eax,eax
            mov eax, dword ptr min
            add eax, ecx                                ;  запись  в  EAX  макси-
мально возможного (в данном диапазоне) элемента массива случайных чисел
            dec eax
            mov ebx,[esi]                               ;  запись  в  EBX  макси-
мальной левой границы

            counter:
                  cmp eax,ebx                           ; если число меньше ле-
вой границы, то ...(см. [*])

                  jl lower
                        push eax
                        push esi                        ;  если  же  больше
или равно, то
```

```
                        mov esi,SUPER_counter         ; помещаем в ESI счёт-
чик для границы
                        mov edx,[esi]                 ;    заносим    его
значение в EDX
                        mov eax,[edi]                 ; помещаем коли-
чество вхождений в свой единичный интервал данного числа
                        add edx,eax                   ; прибавляем это
количество
                        mov [esi],edx                 ;       записываем
обратно в счётчик для НЕединичных интервалов то, что мы получили
                        pop esi
                        pop eax
                        jmp to_previous

                  lower:                              ; [*]
                  sub esi,4                           ;  ...Берём меньшую ле-
вую границу,
                  sub SUPER_counter,4                 ; сдвигаем счётчик, ко-
торый будет считать количество попаданий для неё,
                  mov ebx,[esi]                       ; заносим новую границу
в EBX
                  jmp counter

                        to_previous:
                        dec eax                                ;      здесь
просто уменьшаем рассматриваемое число и указатель на счётчик для единичных
интервалов
                        sub edi,4
                        loop counter

            pop edx
            pop ecx
            pop ebx
            pop eax
            pop esi
            pop edi

            ret
      custom endp
end
```