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 _____ Кодуков А.В.

Преподаватель _____ Ефремов М.А .

Санкт-Петербург
2020

Задание:

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.

Вариант 10: Подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде двух ассемблерных модулей, первый из которых формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат. Это распределение должно выводиться в текстовом виде для контроля. Затем вызывается второй ассемблерный модуль, который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами). Это распределение возвращается в головную программу и выдается как основной результат в виде текстового файла и, возможно, графика.

Выполнение работы:

Были реализованы две процедуры на языке ассемблер.

UNIT – считает распределение чисел по интервалам единичной длины

ARBITARY – считает распределение чисел по заданным интервалам

Они были встроены в проект на C++ с помощью extern “C”.

Тестирование:

```
Распределение случайных чисел по инервалам единичной длины:
№       Лев.гр.         Кл-во
0       0               0
1       1               1
2       2               2
3       3               3
4       4               1
5       5               0
6       6               0
7       7               1
8       8               1
9       9               4
10      10              2
11      11              3
12      12              2
13      13              0
14      14              2
15      15              0
16      16              3
17      17              1
18      18              0
19      19              1
20      20              3
```

```
Распределение случайных чисел по инервалам еденичной длины:
№       Лев.гр.         Кол-во
0       0               24
1       5               25
2       10              28
3       15              23
```

Вывод:

В ходе выполнения работы была организована связь ассемблера с ЯВУ(C++), реализована программа построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

freq.asm

```
.686
.MODEL FLAT, C
.DATA
offsetcountn dd 0

.CODE
PUBLIC C UNIT
;распределение по интервалам единичной длины
UNIT PROC C Number:dword, NumRanDat:byte, CountNumUnit1:dword, Xmin:byte
  PUSH EDI ;сохранение регистров
  PUSH ESI
  MOV EAX, DWORD PTR Xmin ;получение значение xmin
  MOV EDI, Number ;получение адреса массива случайных чисел
  MOV ESI, CountNumUnit1 ;получение адреса массива счетчика чисел
  MOV ECX, DWORD PTR NumRanDat ;получение длины массива случайных чисел

  cycle:
    MOV EBX,[EDI] ; получение числа
    SUB EBX,EAX ; вычесть xmin
    MOV EDX,[ESI+4*EBX] ; получить счетчик этого числа
    INC EDX ; прибавить 1
    MOV [ESI+4*EBX],EDX ; занести в обратно массив
    ADD EDI,4 ; переход к след числу
    LOOP cycle ; повторять пока не прошли весь массив
    POP ESI ;восстановление значений регистров
    POP EDI
  RET
UNIT ENDP

PUBLIC C ARBITARY ;распределение по интервалам произвольной длины
ARBITARY PROC C CountNumUnit1:dword, lenUnit1:byte, LGrInt:dword,
CountNumN:dword ,NInt:byte, Xmin:byte
  PUSH EDI ;сохранение регистров
  PUSH ESI
  MOV EDI,DWORD PTR lenUnit1 ;получение указателя на последний элемент массива
счечика единичной длины
  DEC EDI
  SHL EDI,2
  ADD EDI,CountNumUnit1
  MOV ECX,DWORD PTR lenUnit1 ;получение счетчика
  MOV EAX, DWORD PTR NInt ;получение указателей на последние элементы массивов
  DEC EAX
  SHL EAX,2
  push edi
  MOV ESI, LGrInt
  MOV Edi, CountNumN
  ADD Edi,EAX
  ADD ESI,EAX
  MOV offsetcountn,edi
  pop edi
  ; опрерации перед выполнением цикла
  SUB EAX,EAX
  MOV EAX, dword PTR Xmin
  ADD EAX, ECX
  DEC EAX
  MOV EBX,[ESI]

  cycle:
```

```
    CMP EAX,EBX
    JL falseif
    PUSH EAX
    push esi
    mov esi,offsetcountn
    MOV EDX,[Esi]
    MOV EAX,[EDI]
    ADD EDX,EAX
    MOV [Esi],EDX
    pop esi
    POP EAX
    JMP endif1
  falseif:
    SUB ESI,4
    SUB offsetcountn,4
    MOV EBX,[ESI]
    JMP cycle
    endif1:
    DEC EAX
    SUB EDI,4
  LOOP cycle ;повторять до тех ПОР ПОКА НЕ ПРОЙДЕМ ВЕСЬ МАССИВ
  POP ESI ;восстановление значений регистров
  POP EDI
  RET
ARBITARY ENDP
END
```

main.cpp

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

extern "C" {
void UNIT(int* Number, int NumRanDat, int* CountNumUnit1, int Xmin);
void ARBITARY(int* CountNumUnit1, int lenUnit1, int* LGrInt, int* CountNumN,
              int NInt, int Xmin);
}

void PrintArray(int *Arr, int num) {
  for (int i = 0; i < num; i++)
    std::cout << Arr[i] << " ";
  std::cout << "\n";
}

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, 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 = i;
      j = cur_len;
      while (i < j) {
        LGrInt[j] = LGrInt[j - 1];
```

```
        j--;
       }
       break;
     }
   }
   LGrInt[i] = num;

   return true;
}

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);
}

bool GetInformation(int& NumRanDat, int*& Number, int& Xmin, int& Xmax,
                    int& NInt, int*& LGrInt) {
   int len = 0, i = 0;

   setlocale(LC_ALL, "rus");

   //получение количества чисел
   std::cout << "1.Введите количество случайных чисел, 0 < N < 16384: ";
   std::cin >> NumRanDat;

   while (NumRanDat <= 0 || NumRanDat >= 16384) {
     std::cout << "Ошибка: число должно быть в приведенном диапазоне\n"
               << " Введите количество случайных чисел: ";
     std::cin >> NumRanDat;
   }

   //получение диапозона
   std::cout << "2.Введите диапазон случайных чисел: \n"
             << "От: ";
   std::cin >> Xmin;
   std::cout << "До: ";
   std::cin >> Xmax;

   while (Xmax <= Xmin) {
     std::cout << "Ошибка: правая граница диапозона должна быть больше левой\n"
               << "Введите правую границу: ";
     std::cin >> Xmax;
   }

   Number = new int[NumRanDat];
   if (!Number) {
     std::cout << "Ошибка: не удалось выделить память\n";
     return false;
   }

   GetRandomNum(Number, NumRanDat, Xmin, Xmax);

   //получение количества интервалов
   std::cout << "3.Введите количестно интервалов, но которые разобьется "
             << "диапозон 0 < N < 25: ";
   std::cin >> NInt;

   while (NInt <= 0 || NInt >= 25) {
     std::cout << "Ошибка: количество интервалов не в указанном диапазоне\n"
               << "Введите количество интервалов: ";
```

```
    std::cin >> NInt;
  }

  LGrInt = new int[NInt];

  if (!LGrInt) {
    std::cout << "Ошибка: не удалось выделить память\n";
    return 0;
  }

  //получение интервалов
  std::cout << "4.Выбор интервалов.\n";

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

  return true;
}

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

void PrintResult1(int *Number, int len) {
  int i = 0;
  std::ofstream fout("result1.txt");

  if (!fout.is_open()) {
    std::cout << "Error: can't open file " << '\n';
    return;
  }

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

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

  fout.close();
}

void PrintResult2(int *LGrInt, int *CountNum, int len) {
  int i = 0;
  std::ofstream fout("result2.txt");

  if (!fout.is_open()) {
    std::cout << "Error: can't open file " << '\n';
    return;
  }

  std::cout << "Распределение случайных чисел по заданным интервалам:\n";
  std::cout << "№\tЛев.гр.\t\tКол-во\n";
```

```
  fout << "Распределение случайных чисел по инервалам еденичной длины:\n";
  fout << "№\tЛев.гр.\t\tКол-во\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();
}

int main() {
  int
    NumRanDat = 0,  //количество превдослучайных чисел
    *Number,  //массив чисел
    Xmin = 0,        //левая граница диапозона
    Xmax = 0,        //правая граница диапазона
    NInt = 0,  //количество границ (не в ключай граничный значения)
    *LGrInt,  //массив левых границ
    *CountNumUnit1,  //для интеревалов еденичной длины
    lenUnit1 = 0,
    *CountNumN;  //для интервалов проивольной длины

  if (!GetInformation(NumRanDat, Number, Xmin, Xmax, NInt, LGrInt))  //получить
значения
    return 1;

  PrintArray(Number, NumRanDat);  //вывести случайные числа
  lenUnit1 = Xmax - Xmin + 1;     //выделение памяти
  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);
  UNIT(Number, NumRanDat, CountNumUnit1,
           Xmin);  // распределение по интервалам 1
  ARBITARY(CountNumUnit1, lenUnit1, LGrInt, CountNumN, NInt,
           Xmin);  //распределение по инервалам произвольной длины

  PrintResult1(CountNumUnit1, lenUnit1);  //вывод результата первой процедуры
  PrintResult2(LGrInt, CountNumN, NInt);  //вывод результата второй процедуры

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

  return 0;
}
```