Отчет по лабораторной работе № 14 по курсу
«Фундаментальная информатика»

Студент группы М8О-105Б-21 Минеева Светлана Алексеевна, № по списку 14

Контакты e-mail: svetlana.mineewa2003@yandex.ru

Работа выполнена: «27» ноября 2021 г.

Преподаватель: Вячеслав Константинович Титов каф. 805

Отчет сдан «27» ноября 2021 г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Вложенные циклы с параметрами. Обход и линеаризация матриц.

2. **Цель работы:** Составить программу ввода квадратной матрицы и печати в строку всех её элементов в заданном в варианте порядке следования (обхода).

3. **Задание** (*вариант №*14)**:**

1 7 13 15
10 2 6 14
11 9 3 5
16 12 8 4

4. **Оборудование** (лабораторное):
ЭВМ _____, процессор _____, имя узла сети _____ с ОП _____ Мб,
НМД _____ Мб. Терминал _____ адрес _____. Принтер _____
Другие устройства _____
_____

Оборудование ПЭВМ студента, если использовалось:
Процессор 2,9 GHz 2-ядерный процессор Intel Core i5 с ОП 8 Гб, НМД 500 Гб. Монитор 13,3-дюймовый (2560 x 1600).
Другие устройства _____
_____

5. **Программное обеспечение (лабораторное):**
Операционная система семейства _____, наименование _____ версия _____
интерпретатор команд _____ версия _____
Система программирования _____ версия _____
Редактор текстов _____ версия _____
Утилиты операционной системы _____
_____
Прикладные системы и программы _____
Местонахождение и имена файлов программ и данных_____
_____

Программное обеспечение ЭВМ студента, если использовалось:
Операционная система семейства UNIX, наименование Terminal версия 2.10
интерпретатор команд bash версия 3.2.
Система программирования _____версия _____
Редактор текстов emacs версия 27.2-2
Утилиты операционной системы cat, ls, cp, mv и другие
Прикладные системы и программы _____
_____
Местонахождение и имена файлов программ и данных на домашнем компьютере _____
_____

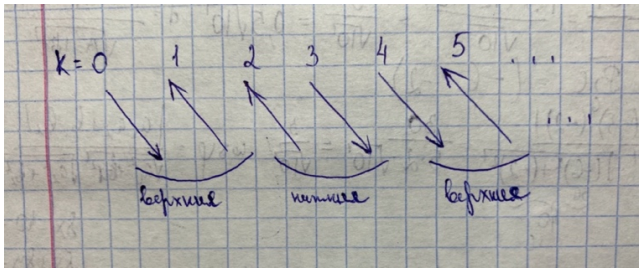
6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Алгоритм рещения задачи:

1) На ввод подаём натуральное число n – количество строк (столбцов) квадратной матрицы;
2) Вводим дополнительные переменные:
    1. i – первый индекс массива (строка);
    2. j – второй индекс массива (столбец);
    3. a[i][j] – двумерный массив;
    4. k – параметр (подсчет пройденных диагоналей);
3) С помощью вложенных циклов заполняем двумерный массив a[i][j] и выводим его на экран;
4) Проверяем: если n равно 1, то выводим на экран единственный элемент матрицы;
5) Присваиваем переменным i и j первоначальное значение -1;
6) Начинаем вложенные циклы с параметром. Изначально параметр k=0, пока k меньше 2*n-3 (количество пройденных диагоналей при проходе матрицы), выполняем следующее, каждый раз увеличивая k на единицу:
    1. Если остаток от деления k на 4 равно 0 или 3 (значит мы двигаемся по диагонали вниз и вправо), то:
        1) Увеличиваем j на единицу;
        2) Начинаем цикл, увеличивающий j на единицу, пока j меньше n, выполняя следующее:
            1 Если i меньше n, то увеличиваем её на единицу, иначе заканчиваем цикл;
            2 Если i меньше n и больше или равно 0, то выводим на экран a[i][j];
    2. Иначе (остаток от деления k на 4 неравно 0 или 3) (значит мы двигаемся по диагонали вверх и влево), то:
        1) Начинаем цикл, уменьшающий j на единицу, пока j больше или равно 0, выполняя следующее:
            1 Если i равно 0, то j увеличиваем на единицу и прекращаем цикл;
            2 Уменьшаем i на единицу;
            3 Если i меньше n и больше или равно 0, то выводим на экран a[i][j];
    3. Если j равно -1, то приравниваем её к нулю;
    4. Если i равно нулю, а j не равно n-1 (значит мы находимся в верхней строке, но не в углах (нахождение в левом верхнем углу не проверяется, так как с этого места мы начинаем обход матрицы и снова в эту точку мы больше не попадём)) , то j равно n-j-1 (*объяснение работы формулы после алгоритма под знаком **), i равно n-1 (выполняем переход на нижнюю строку в еще не пройденный элемент справа). Выводим на экран a[i][j];
    5. Иначе, если i равно n (значит мы находимся в нижней строке, но не в углах (нахождение в них не проверяется, так как значение i при этом будет равно n-1), то j равно (k/2-k/4)*2 (*объяснение работы формулы после алгоритма под знаком ***), i равно 0 (выполняем переход на верхнюю строку в еще не пройденный элемент слева). Выводим на экран a[i][j];
    6. Иначе, если j равно нулю, а i не равно n-1 (значит мы находимся в левом столбце, но не в углах (нахождение в левом верхнем углу не проверяется, так как с этого места мы начинаем обход матрицы и снова в эту точку мы больше не попадём)), то i увеличиваем на единицу (выполняем переход на элемент вниз в столбце). Выводим на экран a[i][j];
    7. Иначе, если j равно n (значит мы находимся в правом столбце, но не в углах (нахождение в них не проверяется, так как значение j при этом будет равно n-1), то j и i уменьшаются на единицу (выполняем переход на элемент вверх по столбцу). Выводим на экран a[i][j];
    8. Если i равно n-1 и j равно 0 (значит мы находимся в левом нижнем углу), то i равно 0, а j равно n-1 (переход в правый верхний угол). Выводим на экран a[i][j];
    9. Иначе если j равно n-1 и i равно 0 (значит мы находимся в правом верхнем углу), то j равно 0, а i равно n-1 (переход в левый нижний угол). Выводим на экран a[i][j].

* Перемещение из верхней строки в нижнюю в еще не пройденный элемент справа:
j = n – j – 1 – допустим, мы находимся в верхней строке в p столбце матрицы из n строк и столбцов, если отсчитывать столбцы слева, то это p+1 столбец по количеству (так как индексация начинается с нуля). Матрицу мы проходим асимметрично относительно сторон, то есть в нижней строке p столбцов справа пройдены, значит нам нужно переместиться в p+1 столбец по количеству с отсчетом, начинающимся с крайнего правого элемента. Для этого берем крайний правый индекс столбцов – это n-1 (так как индексация начинается с нуля) и отнимаем от него значение p (индекс верхнего элемента, в котором находимся).
Проверим для матрицы со значением n = 3. Нам потребуется перемещение из точки – i = 0, j = 1 в точку – i = 2, j = 1. Подставляем значения в формулу j = n – j – 1. Получаем j = 3 – 1 – 1 = 1. Получено верное значение.

** Перемещение из нижней строки в верхнюю в еще не пройденный элемент слева:
j = (k/2 – k/4)*2 - допустим, мы находимся в нижней строке в p столбце матрицы из n строк и столбцов. Параметр k считает количество пройденных диагоналей. Можно заметить, что алгоритм прохода делится по парам диагоналей (две диагонали – проходят по двум элементам, начиная слева, верхней строки, следующие – по двум элементам, начиная справа, нижней строки (нулевая диагональ проходит по одному элементу из верхней и нижней строки)). Значение k/2 – количество полностью выполненных пар (не считая законченной им пары, так как он все равно заканчивает пару диагоналей нижних элементов, подсчет которых нас не интересует), значение k/4 – количество пройденных четвёрок (две пары) (не считая законченной им четверки, так как он все равно заканчивает пару диагоналей нижних элементов, подсчет которых нас не интересует). Если мы вычтем два этих значения, то получим количество пар, идущих по элементам верхней строки. Умножив это значение на два, получим индекс элемента, в который нужно переместиться (по сути это количество пройденных элементов верхней строки, но так как индексация начинается с нуля, то это еще и индекс крайнего, еще не пройденного элемента).
Проверим на матрице со значением n = 4. Нам потребуется перемещение из точки – i = 3, j = 1 в точку – i = 0, j = 2, при этом параметр k равен 3. Подставляем значения в формулу j = (k/2 – k/4)*2. Получаем j = (1 – 0) * 2 = 2. Получено верное значение.



7. Сценарий выполнения работы [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

```
#include <stdio.h>

int main() {
  int a[9][9],n,i,j,k;
  printf ("Input n (<=9) = "); scanf ("%d",&n);
  for (i=0; i<n; i++) for (j=0; j<n; j++) a[i][j]=(i + 1)*10 + j + 1;
  printf ("Sourse matrix:\n");
  for (i=0; i<n; i++){ for (j=0; j<n; j++) printf ("%3d",a[i][j]); printf ("\n");}
  printf ("Result Vector:\n");
  if (n==1) { printf ("%3d",a[0][0]); printf ("\n");}
  i=-1; j=-1;
  for (k=0; k<2*n-3; k++) {
      if ((k%4==0)||(k%4==3)) { j+=1; for (j; j<n; j++) {
         if (i<n) i+=1;
         else break;
         if (i<n && i>=0) { printf ("%4d",a[i][j]);}}}
      else  for (j=j-1; j>=0; j--) {
         if (i==0) {j+=1; break;}
         i-=1;
         if (i<n && i>=0) { printf ("%4d",a[i][j]);}}
      if (j==-1) j=0;
      if (i==0 && j!=n-1) { j=n-j-1; i=n-1; printf ("%4d",a[i][j]);}
      else if (i==n) { i=0; j=(k/2-k/4)*2; printf("%4d",a[i][j]);}
      else if (j==0 && i!=n-1) { i+=1; printf ("%4d",a[i][j]);}
      else if (j==n) { i-=1; j-=1; printf ("%4d",a[i][j]);}
      if (i==n-1 && j==0) { i=0; j=n-1; printf ("%4d",a[i][j]); printf ("\n");}
      else if (i==0 && j==n-1) { i=n-1; j=0; printf ("%4d",a[i][j]); printf ("\n");}}}
```

Тесты:

1) 4
   Вывод: 11 22 33 44 34 23 12 43 32 21 31 42 13 24 14 41

2) 9
   Вывод: 11 22 33 44 55 66 77 88 99 89 78 67 56 45 34 23 12 98 87 76 65 54 43 32 21 31 42 53 64 75 86 97 13 24 35 46 57 68 79 69 58 47 36 25 14 96 85 74 63 52 41 51 62 73 84 95 15 26 37 48 59 49 38 27 16 94 83 72 61 71 82 93 17 28 39 29 18 92 81 91 19

3) 7
   Вывод: 11 22 33 44 55 66 77 67 56 45 34 23 12 76 65 54 43 32 21 31 42 53 64 75 13 24 35 46 57 47 36 25 14 74 63 52 41 51 62 73 15 26 37 27 16 72 61 71 17

4) 2
   Вывод: 11 22 12 21

5) 1
   Вывод: 11

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

```
Last login: Sat Nov 27 11:02:26 on ttys000

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro-MacBook:~ macbookpro$ cat zag.txt
******************************
*   Минеева Светлана Алексеевна   *
*              М8О-105Б-21              *
*      Лабораторная работа №14      *
******************************
MacBook-Pro-MacBook:~ macbookpro$ cat >f1.out
^C
MacBook-Pro-MacBook:~ macbookpro$ cat >lab14.c
#include <stdio.h>

int main() {
  int a[9][9],n,i,j,k;
  printf ("Input n (<=9) = "); scanf ("%d",&n);
  for (i=0; i<n; i++) for (j=0; j<n; j++) a[i][j]=(i + 1)*10 + j + 1;
  printf ("Sourse matrix:\n");
  for (i=0; i<n; i++){ for (j=0; j<n; j++) printf ("%3d",a[i][j]); printf ("\n");}
  printf ("Result Vector:\n");
  if (n==1) { printf ("%3d",a[0][0]); printf ("\n");}
  i=-1; j=-1;
  for (k=0; k<2*n-3; k++) {
      if ((k%4==0)||(k%4==3)) { j+=1; for (j; j<n; j++) {
         if (i<n) i+=1;
         else break;
         if (i<n && i>=0) { printf ("%4d",a[i][j]);}}}
      else  for (j=j-1; j>=0; j--) {
         if (i==0) {j+=1; break;}
         i-=1;
         if (i<n && i>=0) { printf ("%4d",a[i][j]);}}
      if (j==-1) j=0;
      if (i==0 && j!=n-1) { j=n-j-1; i=n-1; printf ("%4d",a[i][j]);}
      else if (i==n) { i=0; j=(k/2-k/4)*2; printf("%4d",a[i][j]);}
      else if (j==0 && i!=n-1) { i+=1; printf ("%4d",a[i][j]);}
      else if (j==n) { i-=1; j-=1;  printf ("%4d",a[i][j]);}
      if (i==n-1 && j==0) { i=0; j=n-1; printf ("%4d",a[i][j]); printf ("\n");}
      else if (i==0 && j==n-1) { i=n-1; j=0; printf ("%4d",a[i][j]); printf ("\n");}}}
^C
MacBook-Pro-MacBook:~ macbookpro$ ls -l |tail -4
-rw-r--r--   1 macbookpro  staff     0 27 ноя 11:04 f1.out
-rw-r--r--   1 macbookpro  staff  1192 27 ноя 11:05 lab14.c
-rw-r--r--   1 macbookpro  staff  2895  7 июл  2020 pslog_20200707_123036.log
-rw-r--r--   1 macbookpro  staff   208  8 окт 16:26 zag.txt
MacBook-Pro-MacBook:~ macbookpro$ cat lab14.c
#include <stdio.h>

int main() {
  int a[9][9],n,i,j,k;
  printf ("Input n (<=9) = "); scanf ("%d",&n);
  for (i=0; i<n; i++) for (j=0; j<n; j++) a[i][j]=(i + 1)*10 + j + 1;
  printf ("Sourse matrix:\n");
  for (i=0; i<n; i++){ for (j=0; j<n; j++) printf ("%3d",a[i][j]); printf ("\n");}
  printf ("Result Vector:\n");
  if (n==1) { printf ("%3d",a[0][0]); printf ("\n");}
  i=-1; j=-1;
  for (k=0; k<2*n-3; k++) {
      if ((k%4==0)||(k%4==3)) { j+=1; for (j; j<n; j++) {
         if (i<n) i+=1;
         else break;
```

```
            if (i<n && i>=0) { printf ("%4d",a[i][j]);}}}
        else  for (j=j-1; j>=0; j--) {
            if (i==0) {j+=1; break;}
            i-=1;
            if (i<n && i>=0) { printf ("%4d",a[i][j]);}}
        if (j==-1) j=0;
        if (i==0 && j!=n-1) { j=n-j-1; i=n-1; printf ("%4d",a[i][j]);}
        else if (i==n) { i=0; j=(k/2-k/4)*2; printf("%4d",a[i][j]);}
        else if (j==0 && i!=n-1) { i+=1; printf ("%4d",a[i][j]);}
        else if (j==n) { i-=1; j-=1; printf ("%4d",a[i][j]);}
        if (i==n-1 && j==0) { i=0; j=n-1; printf ("%4d",a[i][j]); printf ("\n");}
        else if (i==0 && j==n-1) { i=n-1; j=0; printf ("%4d",a[i][j]); printf ("\n");}}}
MacBook-Pro-MacBook:~ macbookpro$ gcc lab14.c -o f1.out
MacBook-Pro-MacBook:~ macbookpro$ ./f1.out
Input n (<=9) = 4
Sourse matrix:
 11 12 13 14
 21 22 23 24
 31 32 33 34
 41 42 43 44
Result Vector:
  11  22  33  44  34  23  12  43  32  21  31  42  13  24  14  41
MacBook-Pro-MacBook:~ macbookpro$ ./f1.out
Input n (<=9) = 9
Sourse matrix:
 11 12 13 14 15 16 17 18 19
 21 22 23 24 25 26 27 28 29
 31 32 33 34 35 36 37 38 39
 41 42 43 44 45 46 47 48 49
 51 52 53 54 55 56 57 58 59
 61 62 63 64 65 66 67 68 69
 71 72 73 74 75 76 77 78 79
 81 82 83 84 85 86 87 88 89
 91 92 93 94 95 96 97 98 99
Result Vector:
  11  22  33  44  55  66  77  88  99  89  78  67  56  45  34  23  12  98  87  76  65  54  43  32  21  31  42  53  64  75  86  97  13  24
35  46  57  68  79  69  58  47  36  25  14  96  85  74  63  52  41  51  62  73  84  95  15  26  37  48  59  49  38  27  16  94  83  72
61  71  82  93  17  28  39  29  18  92  81  91  19
MacBook-Pro-MacBook:~ macbookpro$ ./f1.out
Input n (<=9) = 7
Sourse matrix:
 11 12 13 14 15 16 17
 21 22 23 24 25 26 27
 31 32 33 34 35 36 37
 41 42 43 44 45 46 47
 51 52 53 54 55 56 57
 61 62 63 64 65 66 67
 71 72 73 74 75 76 77
Result Vector:
  11  22  33  44  55  66  77  67  56  45  34  23  12  76  65  54  43  32  21  31  42  53  64  75  13  24  35  46  57  47  36  25  14  74
63  52  41  51  62  73  15  26  37  27  16  72  61  71  17
MacBook-Pro-MacBook:~ macbookpro$ ./f1.out
Input n (<=9) = 2
Sourse matrix:
 11 12
 21 22
Result Vector:
  11  22  12  21
MacBook-Pro-MacBook:~ macbookpro$ ./f1.out
Input n (<=9) = 1
Sourse matrix:
 11
Result Vector:
 11
MacBook-Pro-MacBook:~ macbookpro$
```

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание
1	Дом.	27.11.21	10:00	При n=1 результирующая матрица не выводится	Ошибка исправлена, результат выводится, в код добавлена дополнительная строка: if (n==1) { printf ("%3d",a[0][0]); printf ("\n");}	Проблема заключалась в том, что при n=1 программа не входит в цикл, так как при проверке j оказывается равной n

10. Замечания автора по существу работы

Эта лабораторная работа очень полезна, она отлично развивает мышление и учит программированию на языке Си.

11. Выводы

Я составила программу ввода квадратной матрицы и печати в строку всех её элементов в заданном в варианте порядке следования (обхода).

Недочёты при выполнении задания могут быть устранены следующим образом: больше практиковаться в написании программ на языке Си.

Подпись студента Минеева С.А