ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»

РЕАЛИЗАЦИЯ ЧИСЛЕННЫХ АЛГОРИТМОВ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ СИ

Методические указания
к лабораторным работам

689325

САМАРА
Издательство СГАУ
2006

Составитель *А.А.Тюгашев*

УДК 004.43(075)

Реализация численных алгоритмов на языке программирования Си: метод.указания к лабораторным работам/ сост. *А.А.Тюгашев.*- Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2006. - 22 с.

В методических указаниях на примере выполнения конкретных лабораторных работ демонстрируются принципы работы с массивами и матрицами, сортировки, нахождения сумм, максимумов и средних значений, работы с файлами и структурами данных в интегрированной среде разработки программ (IDE).

Предназначены для студентов радиотехнического факультета, выполняющих лабораторные работы по курсу "Информатика". Настоящие методические указания могут также быть использованы в практической деятельности. Подготовлены на кафедре компьютерных систем.

Печатаются по решению Редакционно-издательского совета Самарского государственного аэрокосмического университета

Рецензент С. В. Востокин

ОБЩИЕ СВЕДЕНИЯ

Язык программирования Си остаётся одним из главных средств разработки профессиональных программ в мире. Язык Си оказал огромное влияние на развитие современных языков и систем программирования. Си стал основой таких современных языков, как C++, C# и Java, которые де-факто являются на настоящее время стандартами разработки сложных программных систем в ИТ-отрасли.

В то же время язык Си, который изначально создавался как язык системного программирования (в частности, именно на нём написано семейство операционных систем UNIX, включая особенно популярную, созданную по модели открытого программного кода Linux), сохраняет свои позиции и как самостоятельный язык программирования, особенно в таких областях, как приложения реального времени и системные программы.

Авторами языка Си являются Брайан Керниган и Деннис Ритчи, работавшие на момент создания языка в фирме Bell корпорации AT&T.

Настоящие методические указания призваны помочь студентам в выполнении лабораторных работ по курсу информатики.

1. ЛАБОРАТОРНАЯ РАБОТА №1

Задание: Написать и отладить в интегрированной среде разработки программ (Integrated Development Environment, IDE) программу, выводящую на экран текстовое сообщение "Здравствуй, Мир!"

Для начала работы необходимо запустить интегрированную среду разработки программ Турбо Си (обычно с помощью ярлыка на рабочем столе Windows), открыть окно редактирования новой программы (подпункт **New** в пункте главного меню интегрированной среды разработки **File**, переход в главное меню осуществляется по нажатию клавиши F10) и набрать текст программы.

Текст программы:

```
#include <stdio.h>

int main()
{
  printf ("Здравствуй, Мир!\n");
  return 0;
}
```

Разберем построчно текст программы. Прежде всего, следует обратить внимание на стиль написания текста программы, в том числе - использование вертикальных и горизонтальных пробелов для отделения логически значимых частей текста друг от друга (отступы образуют при этом специфическую "лесенку").

Правильный стиль необходим для того, чтобы программа была более понятной и ясной для человека - включая и самого автора по прошествии некоторого времени! - потому что с точки зрения языка программирования Си пробелы несущественны (игнорируются).

Удобство восприятия и "прозрачность" программы для человека - одни из главных факторов успеха крупных современных

профессиональных программных проектов, в которых задействуется множество участников и создаются тысячи и десятки тысяч строк программ.

Обратите также внимание на то, что обычно при написании операторов и стандартных функций языка Си принято использовать строчные, а не прописные буквы.

Любая программа на языке Си может содержать предварительную часть - так называемые директивы препроцессора, которые идут в начале текста программы и для отличия от собственно операторов программы начинаются с символа '#' (диез). Препроцессор - это специальная программа, обрабатывающая тексты программ до того, как они передаются транслятору, и осуществляющая некоторые предварительные действия в соответствии с директивами. Одно из главных назначений директив препроцессора - указание того, какие из библиотек (предварительно кем-то разработанных наборов программ, выполняющих полезные функции) будут использованы в программе. Директива #include, в частности, указывает на необходимость включения в программу заголовочного файла stdio.h библиотеки стандартных функций ввода-вывода (**STandarD Input/Output**). Это необходимо для возможности дальнейшего использования библиотечной функции printf.

Следующая строка содержит заголовок основной функции программы - функции main. Любые программы на языке Си являются наборами функций. Термин "функция" здесь имеет не математический, а особый, специфический смысл. Функция определяется как некоторый модуль (часть программы), имеющий в общем случае некоторые аргументы, записываемые в круглых скобках, и способный возвращать некоторое значение. При этом одни функции могут вызывать на выполнение другие, и так далее.

Функции могут располагаться в тексте программы в различном порядке, но система определяет точку входа (начала исполнения) с начала текста функции **main**. Поэтому каждая программа на языке Си обязательно должна включать одну (и только одну!) функцию с именем main. Остальные функции могут иметь произвольные имена в соответствии с правилами написания идентификаторов (имен) в языке Си (то есть могут содержать

латинские буквы и цифры, не содержать в имени пробелов, начинаться с буквы и пр.).

Перед собственно названием функции пишется тип возвращаемого ей значения. В нашем примере перед main написано ключевое слово **int**, это означает, что функция возвращает целое число.

Поскольку функция main - главная функция программы, то её возвращаемое значение подразумевает возвращаемое программой в операционную систему значение. При этом следует знать, что существует соглашение, по которому, в случае, когда программа завершается нормально, без нештатных ситуаций, она возвращает ноль, в противном случае - ненулевое значение. После названия функции main идут круглые скобки без содержания в них каких-то символов, это означает, что наша программа не обрабатывает никаких входных значений (в случае необходимости можно передать в программу набор некоторых значений с использованием командной строки операционной системы).

В последующих строках после заголовка в тесте находятся собственно "тело"(основной текст) функции, в котором содержатся операторы языка Си, то есть действия, которые она выполняет, и вызовы других функций для исполнения наборов действий. Тело функции ограничивается фигурными скобками - { и }. Такая пара скобок называется в языке программирования Си операторными скобками, и ее используют для группировки любого количества операторов, обрабатываемых потом как один оператор в конструкциях языка.

Наша функция вызывает одну функцию - стандартную библиотечную функцию форматного вывода на дисплей - функцию **printf**. Аргументом данной функции является текстовая строка (текстовые строки в языке Си принято заключать в двойные кавычки - " и "). В конце строки "Здравствуй, Мир!" присутствуют символы "\n", это специальная последовательность символов языка, означающая, что после вывода строки на экран надо перейти к новой строке. Вообще, специальные последовательности символов в языке программирования Си начинаются с символа '\'.

Обратите также внимание, что стандартным разделителем в тексте программ на языке Си, разграничивающим отдельные действия внутри функции, является точка с запятой.

Завершается функция оператором **return 0**. Оператор return прекращает выполнение любой функции и возвращает в вызвавшую её функцию (в нашем случае, поскольку функция главная, она вызывается из операционной системы) некоторое значение в качестве результата. В данном примере мы передаем ноль в операционную систему как свидетельство того, что функция завершилась нормальным образом.

Последняя строка программы - закрывающая фигурная скобка, указывающая на конец функции main.

2. ЛАБОРАТОРНАЯ РАБОТА №2

Задание: Написать программу, осуществляющую математические вычисления с использованием стандартных функций математической библиотеки языка программирования Си (по вариантам). Аргументы для производства вычислений вводятся пользователем с клавиатуры. Программа при этом должна проверять принадлежность введенных значений (числа с плавающей точкой) области допустимых значений для данного выражения, и в случае необходимости - выдавать на экран диагностическое сообщение о непринадлежности аргументов области определения.

Варианты:

1) $\frac{1}{z}\cdot\ln\left(\frac{\sin x}{\cos^2 y+1}\right)$; 2) $e^x\cdot\sqrt{\frac{\mathrm{tg}^2 y-2}{|z-3|}}$; 3) $\frac{\ln(z-5y)}{\sqrt{\cos x-\sin y}}$;

4) $\sqrt{\frac{\cos^2 x+\sin^2 y}{\ln 3z}}$; 5) $\frac{\sqrt{\sin x+\mathrm{tg}z}}{\ln(5x-|3y|)}$; 6) $\sqrt{\frac{\sin x}{\cos z}-\frac{\ln 3z}{\mathrm{tg}\,5y}}$.

Справочная информация:

Для вычисления тригонометрических функций используйте стандартные функции математической библиотеки sin(x), cos(x), tan(x), вместо x при этом подставляется аргумент соответствующей функции. Для вычисления модуля используется функция abs(x). e^x может быть рассчитано с помощью exp(x). Квадратный корень извлекается с помощью стандартной функции sqrt(x), натуральный логарифм находится с помощью log(x).

Текст программы
(для первого варианта):

```
#include <math.h>
#include <stdio.h>
```



```
int main()
  {
     float x,y,z,rez;

printf("Программа расчёта значения математического
выражения\n");
printf("Введите три аргумента (вещественные числа)-
>");
scanf("%f %f %f",&x,&y,&z);

    // Проверяем допустимость введённых аргументов
    if (z==0 || sin(x)<=0)
    {
        printf("Вы ввели недопустимые значения ар-
        гументов\n");
        return 1; // Выход с ненулевым признаком
        завершения
    }

rez=(1/z)*log(sin(x)/(cos(y)*cos(y)+1));
printf("Результат расчета =%7.2f",rez);

   return 0;
 }
```

Обратим внимание на то, что программа предварительно, перед тем как вводить исходные данные, выводит на дисплей сообщение о выполняемых ею задачах. Это является "хорошим тоном" при написании программ - программы пишутся для пользователей, которые в общем случае не знают, для каких функций написана программа и какие данные в нее необходимо вводить. Аналогично каждый ввод данных с клавиатуры должен быть сопровожден содержательным приглашением, в котором желательно указывать число и тип вводимых аргументов.

В первой строке объявляются четыре переменные типа "число с плавающей точкой" - **float** в языке Си. Переменные x,y и z используются для хранения значений аргументов, переменная rez - для сохранения значения результата, подсчитанного по формуле

(вообще-то в данном конкретном случае переменная rez является избыточной и добавлена для повышения наглядности программы).

Вообще, в языке Си идентификаторы, то есть имена переменных и функций, представляют собой последовательность букв латинского алфавита, символов подчеркивания и десятичных цифр. Первым символом в идентификаторе обязательно должна быть буква. При этом заглавные и строчные буквы в языке Си различаются, то есть NAME, Name и name - три разных идентификатора. Имена не должны совпадать с операторами языка и названиями стандартных функций. Базовые типы языка Си: int - целое число; float - вещественное число (число с "плавающей точкой"); char - одиночный символ; void - отсутствие типа; double - вещественное число двойной точности.

Перед названием типа могут встречаться так называемые модификаторы, например unsigned для чисел означает, что данное число не может хранить отрицательное значение. Таким образом, если при 16-разрадном целом числе тип int может хранить числа от -32767 до 32768, то unsigned int - от 0 до 65535.

Используются также модификаторы short и long, указывающие соответственно на короткое (занимающее меньше памяти) и длинное (для хранения больших значений) целое. При этом можно просто писать short и long вместо short int и long int.

Каждая используемая в программе переменная должна быть предварительно объявлена с указанием типа, и в дальнейшем она может использоваться для хранения значений только указанного типа. Допустимо объявлять переменные одного типа как через запятую в одной строке, так и в нескольких строках.

Обратите также внимание на правильный синтаксис вызова стандартной функции ввода данных языка Си **scanf**. Форматная строка для ввода трёх вещественных значений выглядит как "%f %f %f" - три значения типа float. Важно отметить то, что между форматными символами находится ровно по одному пробелу. Далее, для ввода в стандартную функцию scanf передаются не сами переменные, а их адреса, в связи с этим при вводе переменных любого типа, кроме строкового (который сам является массивом, другими словами, указателем), перед именем переменной обязательно присутствует знак '&' (амперсанд).

Далее осуществляется проверка на то, что введённые с клавиатуры значения попадают в область определения вычис-

ляемой функции (помним, что знаменатель не может быть равным нулю - деление на ноль запрещено), а также на то, что логарифм определен только на положительных значениях. Знаменатель $\cos^2 y + 1$ проверять на ноль не требуется, поскольку результатом всегда будет положительное число. Поэтому достаточно проверить z на отличие от нуля, и то, что sin(x) будет положительным. Для других вариантов заданий, возможно, проверка допустимости введенных значений окажется более сложной. Есть вероятность, что там окажется целесообразным использовать вспомогательные переменные для хранения промежуточных значений. Для проверки используется сложное условие, включающее в себя логический оператор ИЛИ (две вертикальные черты, ||). Логическое И записывается как "&&", логическое отрицание знаком '!' (восклицательный знак). Обратите также внимание на то, что в языке программирования Си одиночный знак равенства '=' используется для записи оператора присваивания, а в записи условий равенство обозначается как "==". Для записи сравнений используются также знаки: < - меньше; > - больше; <= - меньше или равно; >= - больше или равно. При условии, что введённое значения переменных x или z попадает в недопустимую область, выдается диагностическое сообщение, и происходит выход из программы с ненулевым значением, передаваемым в операционную систему (return 1). В противном случае выполнение программы продолжается, и после расчёта значения выражения, оно печатается с помощью функции printf. При этом применение форматного символа "%7.2f" позволяет отвести на экране под печать результата ровно семь позиций, причем две из них отводится под печать цифр после запятой (в языке Си вместо запятой пользуются десятичной точкой).

Интересным в тексте программы является также использование комментариев. Для улучшения восприятия программы человеком, помимо правильного корректного стиля использования пробелов и отступов, полезно применение комментариев. Комментарии (примечания) в языке Си - произвольный текст (возможно, включающий несколько строк), заключенный между последовательностями символов "/*" и "*/". В применяемой нами при

выполнении лабораторных работ интегрированной среде разработки Турбо Си разрешается также (как и в языке программирования C++) использование однострочных комментариев, при этом комментарий в строке начинается с пары символов "//". Примечания должны носить содержательный характер, то есть, например, оператор s=a*b не должен комментироваться как "присваивание s произведения a*b", вместо этого в данном случае комментарий может быть "вычисляем площадь прямоугольника".

3. ЛАБОРАТОРНАЯ РАБОТА №3
(4 академических часа)

Задание: Написать программу, которая с использованием цикла **do** языка Си вводит с экрана и накапливает в некоторой переменной сумму положительных целых чисел до тех пор, пока не будет введен ноль либо отрицательное число (оно добавляться не должно). Затем с использованием цикла **while** нужно произвести перевод этого числа в двоичную систему счисления (накоплением последовательно остатков от деления на два в специальном массиве), и с использованием цикла **for** языка Си вывести результат (значения из массива) на печать.

Текст программы:

```
#include <stdio.h>

int main()
{
 int a,s,i,j; // s - сумма
 int ost[20]; // массив остатков от деления на два

 printf("Программа перевода суммы положительных
чисел в двоичную систему\n");
// Сначала накапливаем сумму
s=a=0; // начальная инициализация суммы и слагаемого
do
{
 s+=a;
   printf("введите целое положительное число>");
   scanf("%d",&a);
}
while (a>0);

// Теперь переводим в двоичную систему
i=0; // сбрасываем счетчик остатков от деления в ноль
while(s>1)
{
```

```
 ost[i++]=s%2; // сохраняем остатки в массиве
 s=s/2; // уменьшаем число в два раза
 }
 ost[i]=s; // последняя цифра - частное
 // Печатаем результат - число в двоичной системе
 printf("\nСумма в двоичной форме =");
 for (j=i;j>=0;j--) // выводим в обратном порядке
  printf("%d",ost[j]); // Выводим очередной разряд
 printf("\n",s);

 return 0;
}
```

Обратим внимание на то, что для вычисления остатка от деления используется операция языка Си '%', что нумерация элементов массивов в языке Си начинается с нуля, а также на то, что переменная i используется сначала для подсчёта количества разрядов в получаемом двоичном числе (ограничение задано размером массива - 20), а затем как номер старшего разряда двоичного числа в полученном массиве, от которого переменная цикла **for** - j изменяется до нуля и используется в качестве индекса массива. В записи цикла j-- означает, что после окончания каждого повтора цикла for значение переменной j уменьшается на единицу (существуют также операторы инкремента j++, ++j, а также --j). Вообще, в цикле for в скобках сначала записывается действие, выполняемое до начала цикла, затем после символа "точка с запятой" записывается условие продолжения итераций, затем снова после точки с запятой - действие, выполняемое после каждой итерации. В качестве такого действия часто идет приращение (инкремент) или уменьшение (декремент) индекса. Уменьшение и увеличение может быть и на величину, отличающуюся от единицы, в этом случае удобна ещё одна сокращённая форма записи языка Си: a+=2 вместо a=a+2 (используются также сокращенные формы записи -=, *=, /= и др.).

Внимательно также следует разобраться с использованием переменной a в первом цикле (с помощью оператора do).

4. ЛАБОРАТОРНАЯ РАБОТА №4
(4 академических часа)

Задание: Написать программу, которая вводит с клавиатуры двумерный массив (матрицу) вещественных чисел заданного размера, проводит поиск экстремумов или средних значений в строках или столбцах (по вариантам) и выводит результат в виде вектора найденных значений на экран (предварительно вывести на экран саму матрицу для контроля).

Варианты:

1. Найти максимумы в каждом столбце матрицы 3x5.
2. Найти средние значения в строках матрицы 4x3.
3. Найти минимумы в столбцах матрицы 3x4.
4. Найти средние значения в столбцах матрицы 3x3.
5. Найти максимумы в строках матрицы 5x5.
6. Найти минимумы в строках матрицы 4x5.
7. *усложненный - найти максимумы в диагоналях матрицы 5x5.

Текст программы
(для первого варианта)

```
#include <stdio.h>

int main()
{
 int i,j;
 float a[3][5],Max;

 printf("Программа поиска максимумов в столбцах матрицы 3x5\n");

 // Ввод исходной матрицы
 for (i=0;i<3;i++)
```

```
    for (j=0;j<5;j++)
    {
     printf("введите элемент матрицы a[%d][%d]->",i,j);
     scanf("%f",&a[i][j]);
     }

    // Контрольная печать матрицы
    printf("Введенная матрица:\n");
    for (i=0;i<3;i++)
    {
     for (j=0;j<5;j++)
     printf ("%7.2f",a[i][j]); // Отводим семь позиций
     под элемент
     printf("\n"); // Для перехода на новую строку
     }

     // Поиск максимумов в столбцах
     printf("Максимумы:\n");
     for (j=0;j<5;j++)
     {
      Max=a[0][j]; // Сначала Берём в качестве макси-
мума первый элемент
      for (i=1;i<3;i++) // Цикл по элементам столбца
        if (a[i][j]>Max)Max=a[i][j];
      printf("%7.2f",Max); // Печать максимума текущего
      столбца
     }

     return 0;
    }
```


5. ЛАБОРАТОРНАЯ РАБОТА №5
(4 академических часа)

Задание: Написать программу, которая задает при объявлении (в тексте программы) матрицу вещественных чисел заданного размера, проводит сортировку строк или столбцов матрицы (переставляя местами целиком строки или столбцы) в соответствии с заданным критерием (по вариантам) и выводит результат - отсортированную матрицу (предварительно вывести на экран исходную матрицу для контроля).

Варианты:

1. Матрица 5x5, отсортировать столбцы по убыванию по критерию максимальной суммы элементов столбца.
2. Матрица 4x4, отсортировать строки по возрастанию по критерию минимальной суммы элементов строки.
3. Матрица 3x4, отсортировать столбцы по убыванию по критерию среднего значения элементов в столбце.
4. Матрица 5x3, отсортировать строки по возрастанию по критерию среднего значения элементов в строке.
5. Матрица 5x5, отсортировать столбцы по возрастанию минимальных элементов внутри столбца.
6. Матрица 4x3, отсортировать строки по убыванию максимальных элементов внутри столбца.
7. Матрица 5x5, отсортировать столбцы по убыванию по критерию среднего значения элементов столбца.

Текст программы:
(для первого варианта)

```
#include <stdio.h>

// Вспомогательная функция, используется для вывода
матрицы
int print_Mat(float x[5][5])
{
 int i,j;
```

```
    for(i=0;i<5;i++)
    {
    for(j=0;j<5;j++)
      printf("%7.2f",x[i][j]);
    printf("\n");
    }

}

// Основная функция программы - точка входа
int main()
{
 int i,j,jj;
 float z,MaxSum,Sum,a[5][5]= { {1 ,2  ,3  ,4 ,0.5},
                         {2  ,5  ,-1  ,-7  ,100},
                         {5  ,4  ,3  ,2  ,1},
                         {-100 ,11  ,23.81,-5  ,1},
                         {1 ,2 ,3 ,4 ,5} }; //z -буфер
printf ("Программа сортировки столбцов матрицы 5x5
по убыванию максимумов\n");

// Печатаем матрицу до обработки
printf("Исходная матрица:\n");
print_Mat(a); // Передаем матрицу в функцию
print_Mat для печати

// Начало алгоритма сортировки перестановками
for (j=0;j<4;j++) // Цикл с первого столбца до
предпоследнего
  {
   MaxSum=0;
for (i=0;i<5;i++) MaxSum+=a[i][j]; // Находим сумму
элементов текущего столбца-максимума
for (jj=j+1;jj<5;jj++)
{
 Sum=0;
 for (i=0;i<5;i++) Sum+=a[i][jj]; // Сумма столбца
 кандидата
```

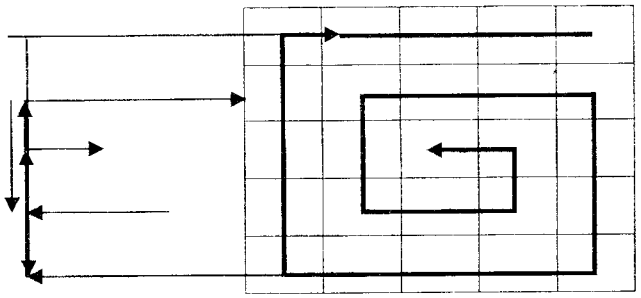


```
 if (MaxSum<Sum)
 {
      MaxSum=Sum; // Новый максимум суммы
      for(i=0;i<5;i++) // Переставляем местами
      столбцы
       {
        z=a[i][j];
        a[i][j]=a[i][jj];
        a[i][jj]=z;
       }
    }
  }
}

// Печать матрицы после сортировки
printf("Результирующая матрица:\n");
print_Mat(a);

return 0;
}
```

Обратите внимание на то, как на языке Си задаются значения элементов двумерной матрицы при объявлении. В программе используется алгоритм сортировки с перестановкой элементов (в качестве элементов здесь фигурируют целые столбцы). Сначала берётся первый столбец и рассматривается в качестве столбца с максимальной суммой элементов. Затем перебираются все остальные столбцы, от второго до пятого, находится сумма элементов для них, и в случае, если она превышает сумму элементов первого столбца, переставляем столбцы (не забывая при этом поменять и значение текущего максимума суммы элементов столбца). Пройдя таким образом все столбцы с первого до предпоследнего, получаем матрицу с отсортированными столбцами. Для печати значений элементов матрицы здесь применяется специально написанная функция - print_Mat. Обратите внимание на правильную запись формальных аргументов при объявлении функции и на правильную передачу в функцию print_Mat матрицы из основной функции.

6. ЛАБОРАТОРНАЯ РАБОТА №6
(4 академических часа)

Задание: "Спиралька". Вывести на экран значения элементов заданной при объявлении матрицы вещественных чисел 5x5, по спирали (см. рисунок).



Текст программы:
(один из возможных вариантов)

```
#include <stdio.h>

int print_Mat(float x[5][5])
{
 int i,j;

 printf ("\n\n\n\n\n");
 for(i=0;i<5;i++)
 {
  for(j=0;j<5;j++)
    printf("%7.2f",x[i][j]);
  printf("\n");
 }
}

int main()
```



```
{
 int i,j,NN; // NN используется как текущая длина
 "плеча"  спирали
 float a[5][5]= { {1,2,3,4,0.5},
                 {2,5,-1,-7,100},
                  {5,4,3,2,1},
                 {-100,11,23.81,-5,1},
                  {1,2,3,4,5} };

// Печать исходной матрицы
print_Mat(a);

// Начало обхода элементов по спирали
for (NN=5;NN>1;NN--)
{
  for (j=(5-NN);j<NN;j++)
   printf("%5.2f ",a[5-NN][j]);
  --j;
  for (i=(5-NN)+1;i<NN;i++)
   printf("%5.2f ",a[i][j]);
  --i;
  for (j=NN-1-1;j>=5-NN;j--)
   printf("%5.2f ",a[i][j]);
  ++j;
  for (i=NN-1-1;i>5-NN;i--)
   printf("%5.2f ",a[i][j]);
 }
}
```

Обратите внимание на то, как отнимаются значения от длины плеча спирали (-1-1 написано именно для того, чтобы подчеркнуть особенности алгоритма). Так же, как и в предыдущей лабораторной работе, используется функция печати матрицы print_Mat. В профессиональном программировании это один из основных приемов - повторное использование написанного кода. При замене всех вхождений числа 5 на переменную N данная программа становится универсальной и может применяться для вывода "по спирали" значений элементов матрицы размером NxN.

Учебное издание

РЕАЛИЗАЦИЯ ЧИСЛЕННЫХ АЛГОРИТМОВ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ СИ

Методические указания к лабораторным работам

Составитель ***Тюгашев Андрей Александрович***

Редактор Л. Я. Чегодаева
Компьютерная верстка О. А. Ананьев

Подписано в печать 18.10.2006 г. Формат 60x84 1/16.
Бумага офсетная. Печать офсетная.
Усл.печ.л. 1,4. Усл.кр.- отт. 1,5. Уч.- изд.л. 1,5.
Тираж 100 экз. Заказ 64. Арт. С-84/2006.

Самарский государственный аэрокосмический университет. 443086 Самара, Московское шоссе, 34.

Изд-во Самарского государственного аэрокосмического университета. 443086 Самара, Московское шоссе, 34.