Уральский федеральный университет

Функции

Лекция 14 - 15

Иллюстративный материал к лекциям по алгоритмизации и программированию

Автор Саблина Н.Г. 2016 г.

Уральский федеральный университет

Содержание

Понятие функции

- ***Функции*** - это самостоятельные единицы программы, предназначенные для решения конкретных подзадач, обычно повторяющиеся несколько раз.
- Перед использованием функция должна быть объявлена
- Все функции в языке Си – глобальные, т.е. функция не может быть объявлена внутри другой функции
- В Си можно объявить функцию с помощью прототипа, т.е. заголовка функции, а полное ее описание сделать после функции main()

Объявление функции

тип <имя функции> (список параметров)

{

тело функции

}

- Функция может возвращать результат
- Тип определяет тип значения, которое возвращает функция
- Если тип не указан, то по умолчанию функция возвращает целое значение (типа **int**).

Список параметров функции

- перечень типов и имен параметров, разделенных запятыми.
- Функция может не иметь параметров, но круглые скобки необходимы в любом случае.

*Пример **правильного** списка параметров:*

f (int x, int y, float z);

Пример неправильного списка параметров:

f (int x, y, float z);

Оператор *return*

- вызывает немедленный выход из текущей функции и возврат в вызывающую функцию.
- используется для возврата значения функции.
- в теле функции может быть несколько операторов ***return***, но может не быть ни одного.
- если нет ***return,*** возврат в вызывающую программу происходит после выполнения последнего оператора тела функции.

Пример функции 1. Возведение неотрицательного числа *a* в натуральную степень *b*

```
float step (float a, int b)
{ int  i;  float  s;
if(a<0) return (-1); /* основание отрицательное */
s=1;
for ( i=b; i; i--) s*=a;          //s=s*a;
return s;
}
```

Эта функция возвращает значение -1, если основание отрицательное, и a^b*, если основание неотрицательное.*

Пример программы, использующей функцию step ():

```
#include <stdio.h>
float step (float , int ) ; //прототип функции
main()
{float x; int y;
printf (" \nВведите основание степени x=");  scanf ("%f", &x);
printf (" Введите показатель степени y=");     scanf ("%d", &y);
if (step (x,y) +1) printf(" x в стпени y=%f\n", step(x,y));
else printf(" основание отрицательно \n");
}
float step(float a, int b)
{
…   return s;
}
```

Пример функции 2. функция для нахождения наибольшего из двух целых чисел

a)
```
int max(int a, int b)
{int m;
if(a>b) m=a; else   m=b;
return m;}
```

b) без использования дополнительной переменной:

```
int  max(int a, int b)
{ if(a>b) return a; else return b;}
```

c) короткий if (без ветви else) :

```
int max(int a, int b)
{ if(a>b) return a;  return b;}
```

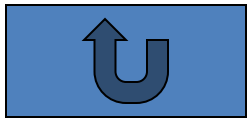
d) с использованием условной операции:

```
int  max(int a, int b){ return (a>b)? a: b;}
```

Формальные и фактические параметры функции

- ***Формальные параметры*** - это переменные, объявленные при описании функций как ее аргументы.

- ***Фактические параметры*** - это параметры, с которыми функция вызывается для выполнения



Область видимости переменных

- ***Область действия*** (видимости) ***переменной*** - это правила, которые устанавливают, какие данные доступны из данного места программы.

- С точки зрения области действия переменных различают три типа переменных:
  - *глобальные*
  - *локальные*
  - *формальные параметры*.

Локальные переменные

- Это переменные, объявленные внутри функции.
- Локальная переменная доступна внутри блока, в котором она объявлена.
- Локальная переменная существует пока выполняется блок, в котором эта переменная объявлена. При выходе из блока эта переменная (и ее значение) теряется.

Глобальные переменные

- Это переменные, объявленные вне какой-либо функции.
- Могут быть использованы в любом месте программы, но перед их использованием они должны быть объявлены.
- Область действия глобальной переменной - вся программа.

Недостатки использования глобальных переменных

- они занимают память в течение всего времени работы программы;
- делает функции менее общими и затрудняет их использование в других программах;
- использование внешних переменных делает возможным появление ошибок из-за побочных явлений. Эти ошибки, как правило, трудно отыскать.

Формальные параметры

- используются в теле функции так же, как локальные переменные.

- Область действия формальных параметров - блок, являющийся телом функции.

Параметры-значения

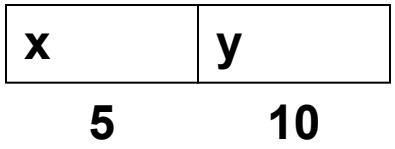
- Все аргументы функции передаются по значению
- При вызове функции в стеке выделяется место для формальных параметров функции, и туда заносится значение фактического параметра, т. е. значение параметра при вызове функции.
- Далее функция использует и меняет значения в стеке. При выходе из функции измененные значения параметров теряются.
- В языке С вызванная функция не может изменить переменные, указанные в качестве фактических параметров в функции при обращении к ней.

Пример 3. Передача данных по значению (1)

```
void swap(int a, int b)
{
int tmp;
tmp = a;
a = b;
b = tmp;
}
```

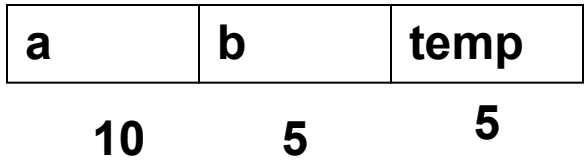
```
#include <stdio.h>
//прототип функции
void swap(int a, int b);
void main( )
{
int x = 5, y = 10;
printf (“Вначале x = %d и y = %d\n”, x, y);
swap(x, y);
printf (“Теперь x = %d и y = %d\n”, x, y);
}
void swap(int a, int b)    //реализация
     функции
{    … }
```

Пример 3. Передача данных по значению (2)




```
main ()
{
int x = 5, y = 10;
printf (“Вначале x = %d и y =
%d\n”, x, y);

swap(x, y);
printf (“Теперь x = %d и y = %d\n”,
x, y);
}
```




```
void swap(int a, int b)
{
int tmp;
tmp = a;
a = b;
b = tmp;
}
```

Параметры - ссылки

- Для возможности изменения внутри функции значений переменных, являющихся параметрами этой функции, необходимо передавать в функцию не значения этих переменных, а их адреса

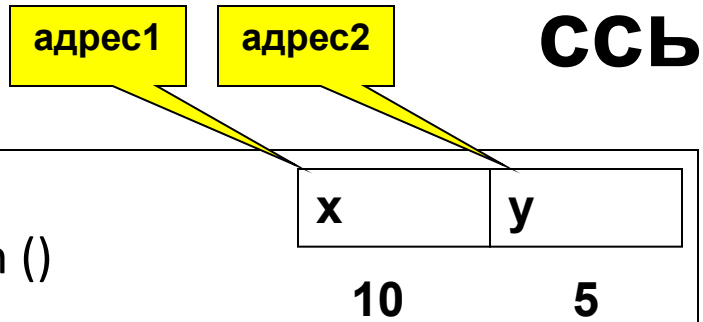
Пример 3а. Передача данных по ссылке (1)

```
void swap1(int *a, int *b)
{
int tmp;
tmp = *a;
*a = *b;
*b = tmp;
}
```

Пример 3а. Передача данных по ссылке (2)

```
#include <stdio.h>
void swap1(int* a, int* b); //прототип функции
void main( )
{
int x = 5, y = 10;
printf (“Вначале x = %d и y = %d\n”, x, y);
swap1 (&x, &y);
printf (“Теперь x = %d и y = %d\n”, x, y);
}
void swap1 (int* a, int* b)    //реализация функции
{    … }
```

Пример 3а. Передача данных по ссылке (3)




```
main ()
{
int x = 5, y = 10;
printf ("Вначале x = %d и y = %d\n", x, y);

swap1 (&x, &y);
printf ("Теперь x = %d и y = %d\n", x, y);
}
```

a	b	temp
адрес1	адрес2	5

```
void swap1 (int* a, int* b)
{
int tmp;
tmp = *a;
*a = *b;
*b = tmp;
}
```

Передача массивов в функцию

- Если в качестве **аргумента функции** используется **массив**, то необходимо в качестве формального параметра передать **адрес начала массива**.
- **Адрес любого** другого **элемента** массива можно вычислить **по** его **индексу** и **типу** элементов массива.

Пример 4. Функция поиска максимального значения в массиве (1)

```
int max_m (int *a, int N)
{int  im, j;
im = 0;
for(j = 1; j < N; j++)
if(a[j] > a[im])  im = j;
return a[im];
}
```

Пример 4. Функция поиска максимального значения в массиве (2)

```
#include <stdio.h>
int max_m(int *a, int N);
void main( )
{int N, i, mas[100],  max;
puts("Введите размер массива, но не более 100");
scanf("%d", &N);
for(i = 0; i < N; i++) mas[i] = random(100);
max=max_m (mas, N);
printf("Максимальный элемент  равен %d",  max);
}
//реализация функции max_m
int max_m(int *a, int N) {  …  }
```

Передача матриц в функцию через параметры

- Для передачи матрицы в функцию в качестве параметров нужно указать
  - адрес начала вспомогательного массива указателей на начала срок матрицы
  - размерность матрицы: количество строк и столбцов

Пример 5. Поиск строк матрицы, не содержащих нулей

Имеются две **матрицы A и B** натуральных чисел размерностью Na x Ma, Nb x Mb соответственно.

Вычислить произведение элементов в тех строках матриц A и B, которые не содержат нулевых элементов.

Постановка задачи

Исходными данными для этой задачи являются:

Na, Nb – количество строк в матрицах,

Ma, Mb – количество столбцов в матрицах; целые числа, вводятся с клавиатуры.

A, B – матрицы, заполняются случайными числами в ходе выполнения программы.

Выходными данными являются произведения элементов выбранных строк матриц.

Метод решения задачи

Решение нашей задачи можно разделить на несколько подзадач:

а) создание и заполнение матрицы заданного размера;

б) вывод матрицы на экран;

в) поиск строки, не содержащей нулей;

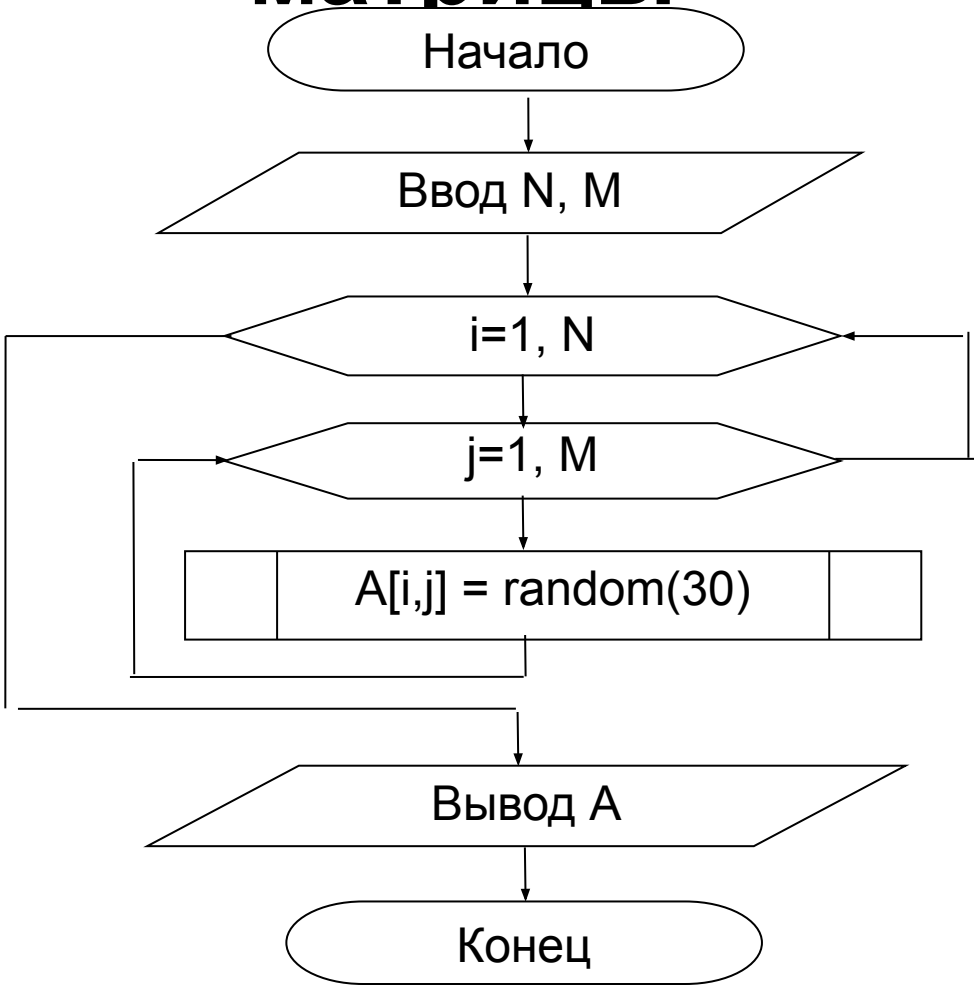
г) вычисление произведения элементов найденной строки.

Каждую из этих подзадач оформим в виде отдельной функции.

Создание и заполнение матрицы (функции InitMatr)

- Матрица создается динамически.
- Заполняется с помощью датчика случайных чисел с параметром
- В качестве параметров передадим в эту функцию размерность матрицы: M и N.
- Память под матрицу выделятся динамически, с использованием дополнительного массива указателей на начала строк матрицы.
- Сама функция возвращает адрес начала массива указателей на начала строк матрицы.
- Функция вызывается дважды: для матрицы A и для матрицы B

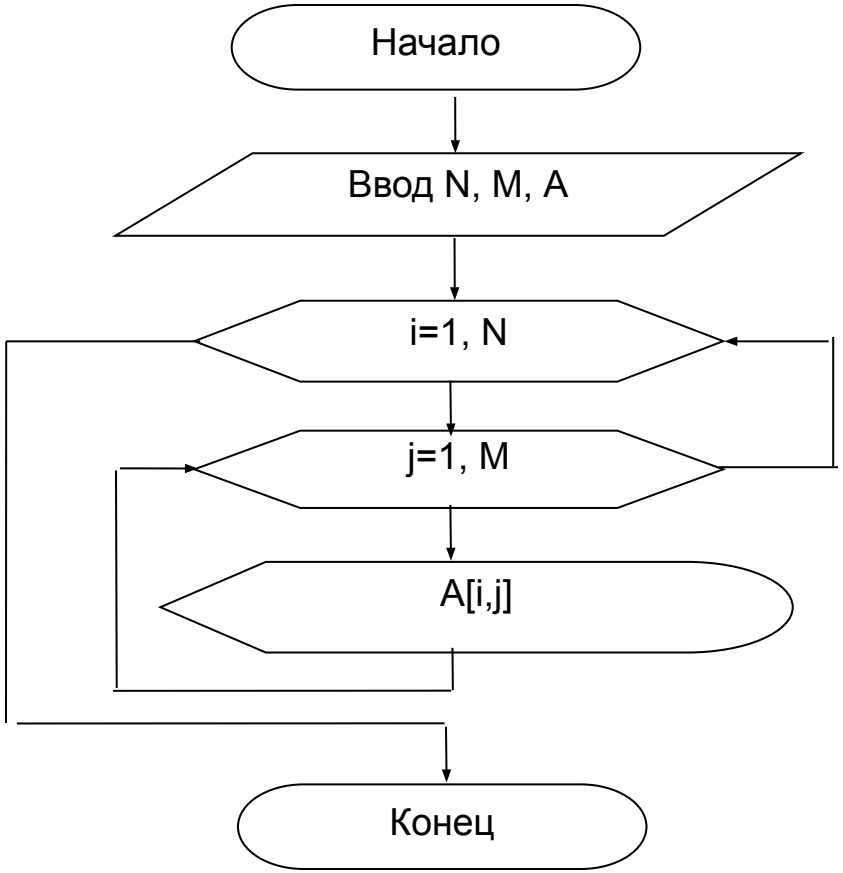
Схема алгоритма формирования матрицы



Вывод матрицы на экран (функция OutMatr)

- Через параметры в эту функцию будут передаваться:
  - размерность матрицы (передается по значению),
  - сама матрица (передается по ссылке на начало массива указателей).
- Функция вызывается дважды: для матрицы A и для матрицы B

Схема алгоритма вывода матрицы на экран



Поиск строки без нулей и вычисление произведения элементов в строке

- удобно оформить в виде отдельных функций. В качестве параметров передавать в эти функции
  - строку (одномерный массив)
  - ее размер (количество элементов в строке).
- Функция поиска нулей возвращает 0, если нули в строке есть, и 1 , если нулей нет.
- Функция вычисления произведения возвращает значение произведения.

Алгоритм поиска строки без нулей

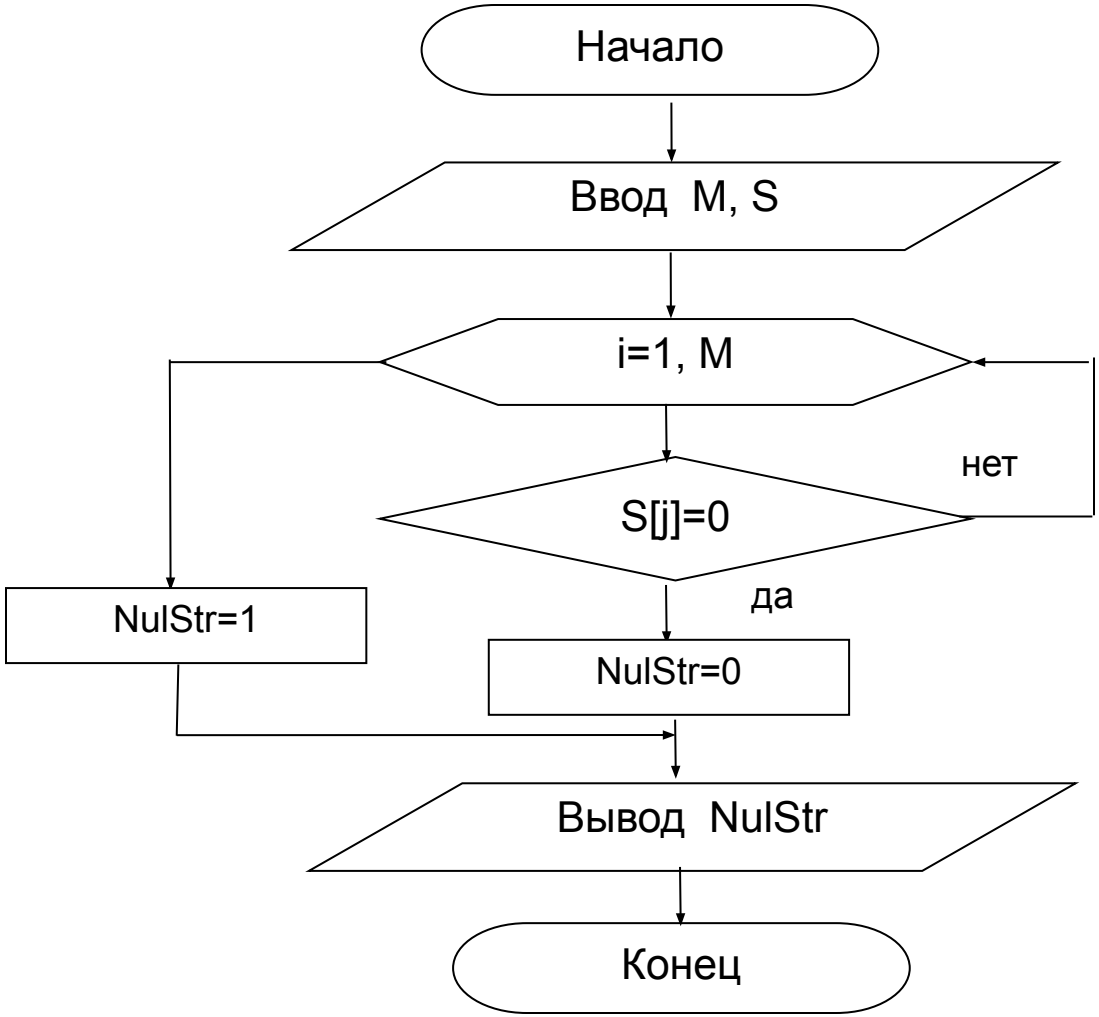


Схема алгоритма вычисления произведения элементов строки матрицы (функция PrStr)

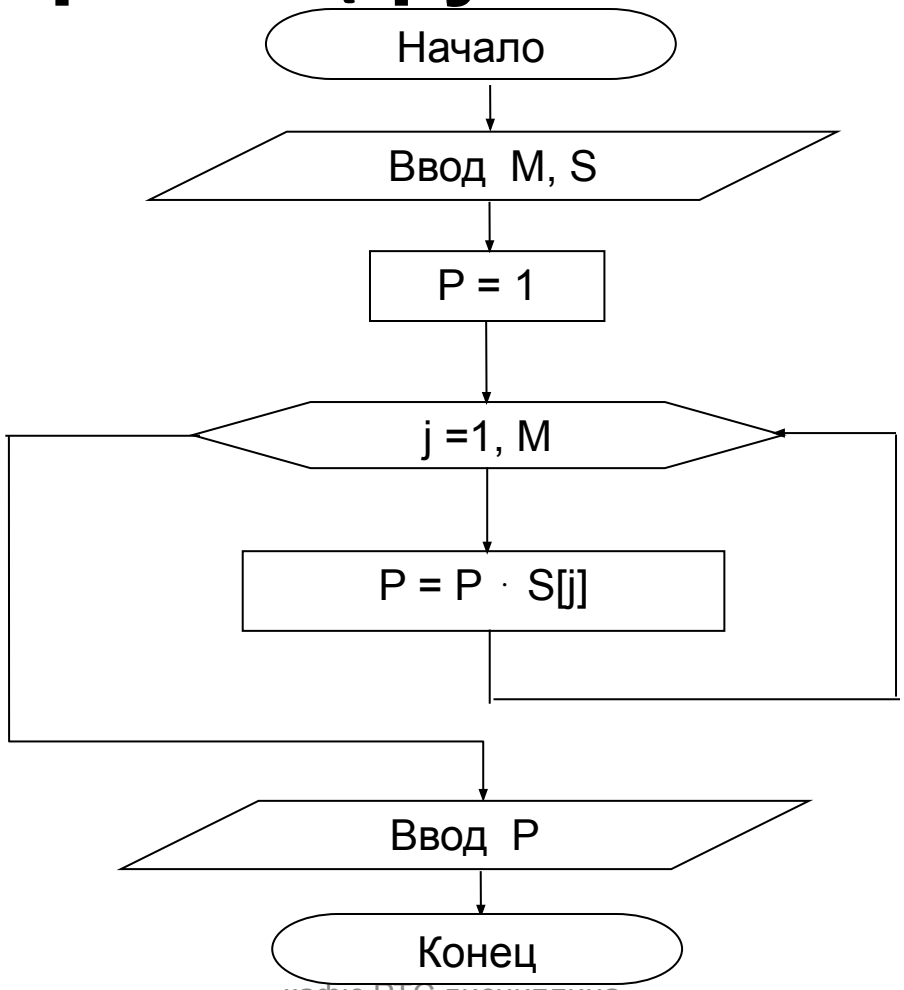


Схема главной функции (1)

Схема главной функции (2)

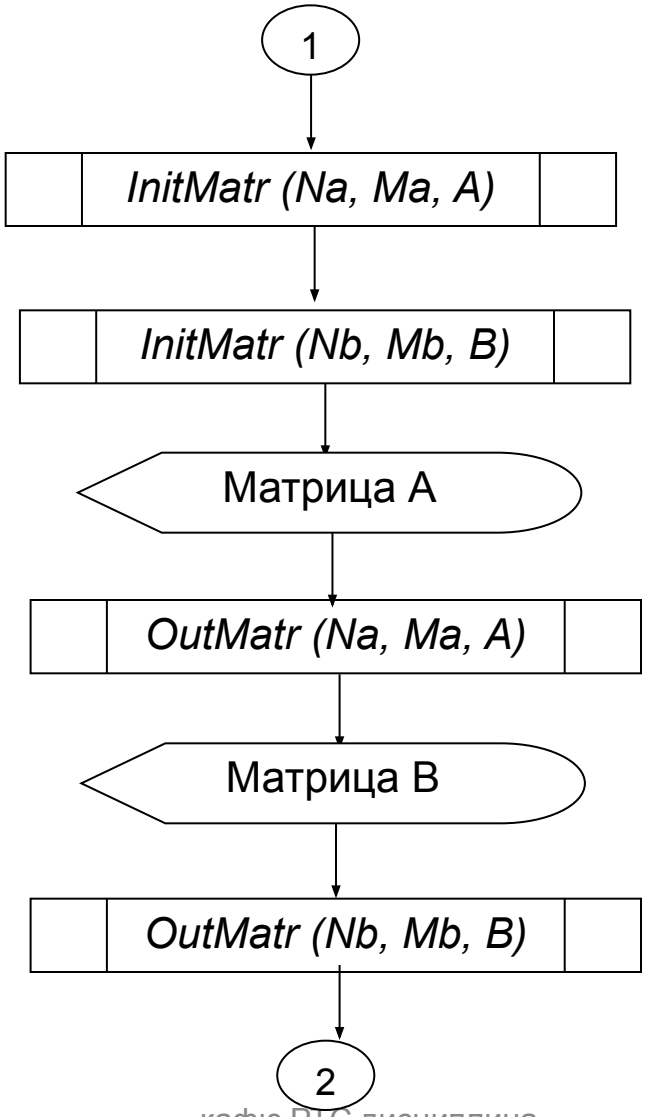


Схема главной функции (3)

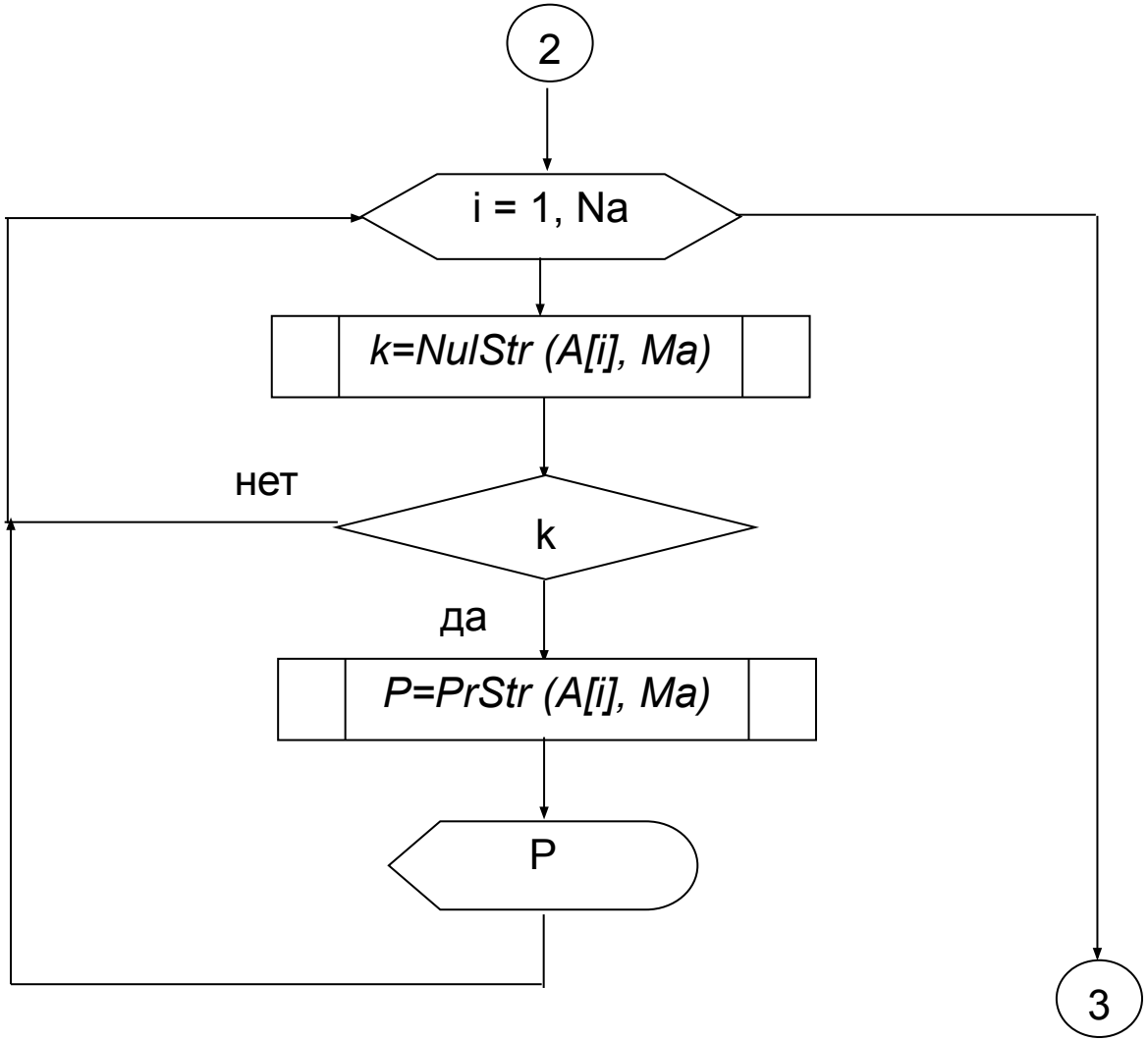
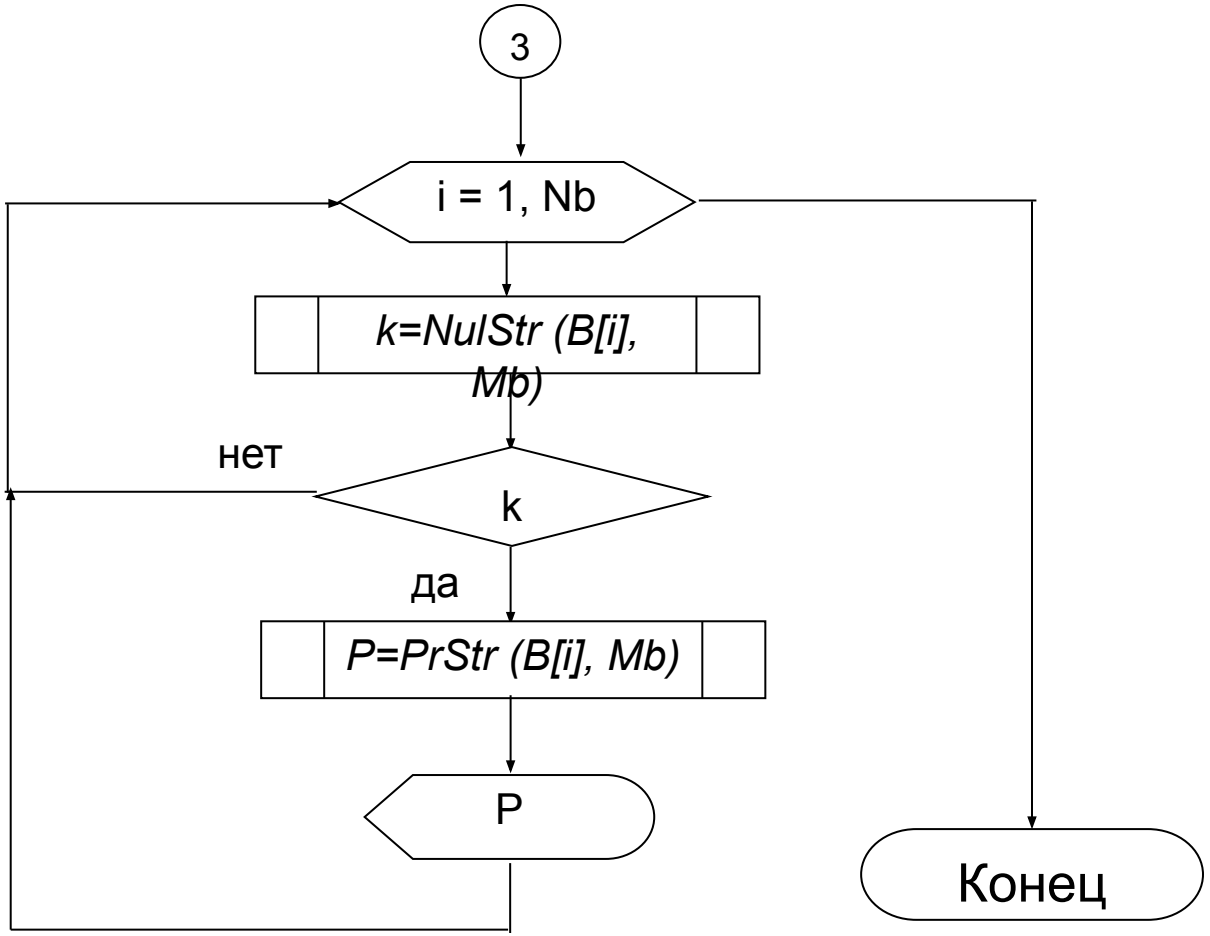


Схема главной функции (4)



Исходный текст программы (1)

```
#include <stdio.h>
#include <stdlib.h>
//прототипы функций
int ** InitMatr (int N,int M);
 //N,M -размерность матрицы; параметры-значения
//функция возвращает адрес созданной матрицы
void OutMatr(int **A, int N, int M);
//N,M -размерность матрицы;
//A - матрица, передается через адрес массива указателей
//на начала строк
```

Исходный текст программы (2)

```
//---Функция определения в строке нулевых элементов ----
int NulStr (int *S, int M);
//S - адрес строки массива, M - кол-во элементов в строке
//---Функция вычисления произведения элементов в строке--
long int PrStr(int * S, int M);
```

Исходный текст программы (3)

```
//--------Головная программа --------
main()
{ int ** A, **B ; //адреса исходных матриц}
int   Na,Ma,Nb,Mb; // размерности матриц
//ввод исходных данных
printf("\nВведите размеры матриц  A и B\n");
printf("Количество строк в матрице A Na=");scanf("%d",&Na);
printf (" Количество столбцов в матрице A Ma="); scanf("%d",&Ma);
printf (" Количество строк в матрице B Nb=");scanf("%d",&Nb);
printf (" Количество столбцов в матрице B Mb="); scanf("%d",&Mb);
```

Исходный текст программы (4)

```
randomize();

A=InitMatr(Na,Ma); //Заполнение матрицы A

B=InitMatr(Nb,Mb); // Заполнение матрицы  B


//Вывод исходных матриц на экран

printf ("\nМатрица  A\n");  OutMatr(A,Na,Ma);

printf("\nМатрица  B\n");  OutMatr(B,Nb,Mb);
```

Исходный текст программы (5)

```
//printf ('Произведения элементов в строках без нулей ');
printf ("\n В матрице  A:\n" );
for (int j=0; j<Na ; j++)  if (NulStr (A[j], Ma))
printf ( "\nСтрока %d P=%d", j, PrStr(A[j], Ma));

printf ("\n В матрице  B:\n" );
for ( j=0; j<Nb ; j++)  if (NulStr (B[j], Mb))
printf ( "\nСтрока %d P=%d", j, PrStr(B[j], Mb));
}
```

Исходный текст программы (6)

```
//---------- функция заполнения матрицы ----------------}
int ** InitMatr (int N,int M)
{int ** A;
A=new int* [N];
  for (int i=0 ; i< N; i++) A[i]=new int [M];
for (i=0 ; i< N; i++)
  for (int j=0; j<M; j++)  A[i][j]=random(10);
return A;
}
```

Исходный текст программы (7)

```
//----функция вывода матрицы на экран------}


void OutMatr(int **A, int N, int M)
{
for (int i=0 ; i< N; i++) {
  for (int j=0; j<M; j++) printf ("%4d ", A[i][j]);
  printf("\n");}
}
```

Исходный текст программы (8)

```
//{--Функция определения в строке нулевых элементов ----}


int NulStr (int *S, int M)
{
for (int i=0; i< M; i++)
    if (S[i]==0) return 0;
return 1;
}
```

Исходный текст программы (9)

```
//--Функция вычисления произведения элементов в строке—


long int PrStr(int *S, int M)

{

long int P=1;

for (int i=0; i< M ; i++)  P=P*S[i];

return P;

}
```

Задания для самостоятельного решения

Вариант 1.

Описать функцию *next*() без параметров, которая считывает с клавиатуры первый символ, отличный от пробела, и объявляет его своим значением. Использовать эту функцию для подсчета количества отличных от пробела символов вводимого текста.

Вариант 2.

Даны длины *a, b, c* сторон некоторого треугольника. Найти медианы треугольника, сторонами которого являются медианы исходного треугольника. Длина медианы, проведенной к стороне a, равна

$$0.5\sqrt{2b^2+2c^2-a^2}$$

Вариант 3.

Описать рекурсивную функцию *root*(*f,a,b,eps*), которая методом деления отрезка пополам находит с точностью *eps* корень уравнения *f*(*x*)=0 на отрезке [*a,b*]. (Считать, что *eps* > 0, *a* > *b*, *f*(*a*)**f*(*b*) < 0). Найти с ее помощью один корень уравнения *sin*(*x*)=0.5.

Вариант 4.

Разработать процедуру вычисления $y = \sqrt[k]{x}$ заданной точностью eps по следующей итерационной формуле:

$$y_0 = 1;\ y_{n+1} = y_n + \left(x / y_n^{k-1} - y_n \right) / k,\ n = 0,1,2,\ldots$$

Вариант 5.

Разработать функцию поиска подстроки в массиве символов. При успешном поиске возвращать позицию подстроки относительно начала массива.

Вариант 6.

Даны три целые матрицы размером 9*4. Напечатать ту из них, в которой больше нулевых строк. Для подсчета количества нулевых строк, используйте функцию.

Вариант 7.

Написать функцию, равномерно дополняющую строку пробелами до определенной длины и с ее помощью отформатировать простейший текст.

Вариант 8.

Даны три вещественных матрицы 4*4. Напечатать квадрат матрицы с наименьшим следом.

Вариант 9.

Ввести с клавиатуры матрицу 4*4 и подсчитать ее определитель. Процедуру подсчета определителя оформить в виде функции.

Вариант 10.

Написать функцию перевода строчных букв, введенного с клавиатуры текста, в прописные и обратно.

Контрольные вопросы по теме работы

- Что такое функция?
- Что такое прототип функции?
- Чем отличаются глобальные и локальные переменные?
- Что такое формальные параметры?
- Как передать аргументы в функцию из головной программы по значению?
- Как передать аргументы в функцию из головной программы при помощи указателей?
- Каким образом функция возвращает значения в программу?

- Для чего нужен оператор *return*?
- Как передать массив функции?
- Где должна быть описана функция и где может располагаться тело функции?

Итоги

Рассмотренные вопросы:

- Функции
- Прототип функции
- Типы переменных
- Формальные параметры

Библиографический список

- Подбельский В.В., Фомин С.С. Курс программирования на языке Си: учебник. М.: ДМК Пресс, 2012. – 384 с.
- Павловская Т.А. C/C++. Программирование на языке высокого уровня: учебник для студентов вузов, обучающихся по направлению "Информатика и вычисл. техника" СПб.: Питер, 2005. - 461 с.
- Павловская Т. А., Щупак Ю. А. С++. Объектно-ориентированное программирование. Практикум. Практикум. — СПб.: Питер, 2006. — 265 с: ил.
- Березин Б.И. Начальный курс C и C++ / Б.И. Березин, С.Б. Березин. - М.: ДИАЛОГ-МИФИ, 2001. - 288 с
- Каширин И.Ю., Новичков В.С. От С к С++. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2012. – 334 с.

Автор:

Саблина Наталья Григорьевна

Ст. преподаватель

каф. РТС УрФУ