Основы программирования (на языке Си)

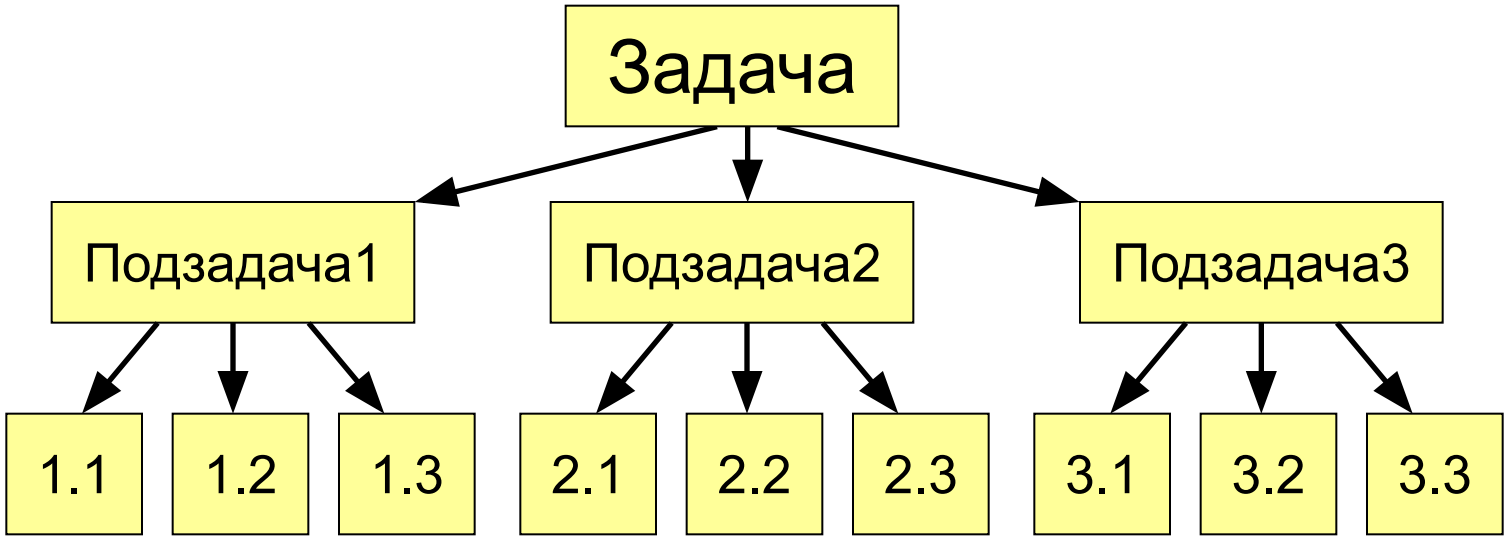
Тема 16. Процедуры

Процедуры

Процедура – это вспомогательный алгоритм, который предназначен для выполнения некоторых действий.

Применение:

- выполнение одинаковых действий в разных местах программы
- разбивка программы (или другой процедуры) на подзадачи для лучшего восприятия



Процедуры

Задача:

Вывести на экран целое число X в целой положительной степени N

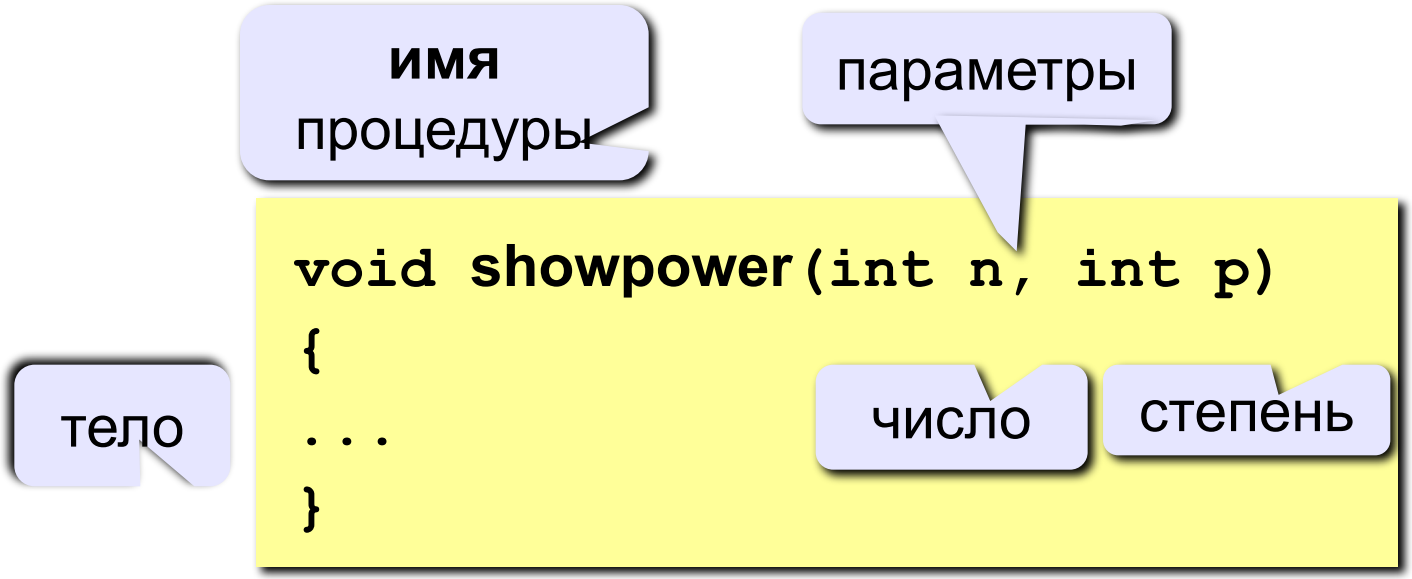
Алгоритм решения:

1) Расчет $x^N = x*x*x*x*\ldots x;$ (N)
2) Вывод x^N на экран (**printf**)

Процедуры

Порядок разработки:

- выделить одинаковое или похожее (вывод числа)
- найти в них **общее** (операция возведения в степень) и **отличия** (число и показатель степени)
- записать отличия в виде неизвестных переменных, они будут **параметрами** процедуры



`void` – «пустой» (некоторые действия)

Процедуры

формальные параметры

```
void showpower( int n, int p)
{     int k,i;
   k=1;
   for (i=0;i<p;i++)
      k*=n;
   printf("%8d",k);
}
```

тело процедуры

«**Формальные параметры**» могут изменяться, заранее неизвестны (обозначаются именами, как переменные).

Программа

формальные параметры

```
#include <conio.h>
#include <stdlib.h>
void showpower( int n, int p)
{
   ...
}

main()
{
   int a=10;
   int b=2;
   showpower(a,b);
   getch();
}
```

процедура

вызовы процедуры

фактические параметры

Процедуры

Особенности:

- *обычно* процедуры расположены **выше** основной программы
- в заголовке процедуры перечисляются **формальные** параметры, они обозначаются именами, поскольку могут меняться

```
void showpower( int n, int p)
```

- при вызове процедуры в скобках указывают **фактические** параметры (числа или арифметические выражения) **в том же порядке**

```
showpower ( 10, 2);
```

n p

Процедуры

Особенности:

- для каждого формального параметра в заголовке процедуры указывают его **тип**

```
void A ( int x, float y, char z ) { ... }
```

- внутри процедуры параметры используются так же, как и переменные
- в процедуре можно объявлять дополнительные **локальные переменные**, остальные процедуры не имеют к ним доступа

```
void A ( int x, float y, char z )
 {
   int a2, bbc =
         345;
 ...
 }
```

локальные переменные

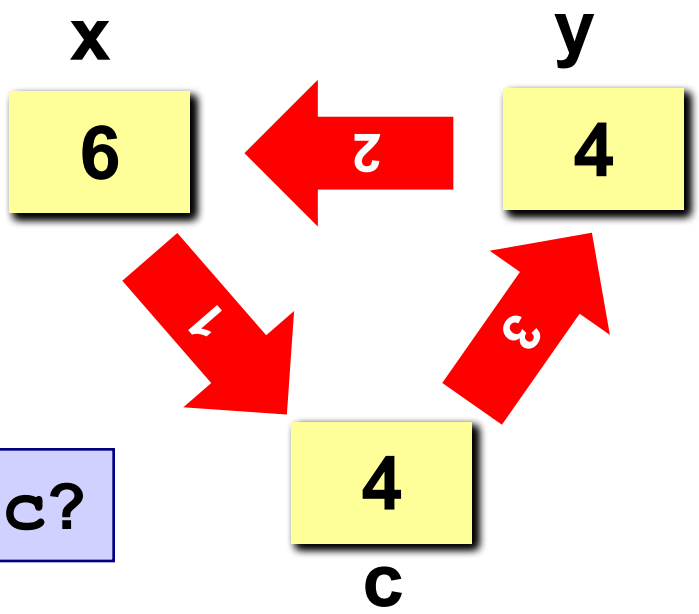
Как поменять местами?

Задача: поменять местами содержимое двух чашек.

Задача: поменять местами содержимое двух ячеек памяти.

```
x = y;
y = x;
```

```
c = x;
x = y;
y = c;
```



Можно ли обойтись без c?

Параметры-переменные

Задача: составить процедуру, которая меняет местами значения двух переменных.

Особенности: надо, чтобы изменения, сделанные в процедуре, стали известны вызывающей программе.

```
void Swap ( int a, int b )
{
int c;
c = a; a = b; b = c;
}
main()
{
  int x = 1, y = 2;
  Swap ( x, y );
  printf ( "x = %d,  y = %d", x, y );
}
```

эта процедура работает с **копиями** параметров

x = 1, y = 2

Параметры-переменные

```
void Swap ( int & a, int & b )
{
 int c;
 c = a; a = b; b = c;
}
```

параметры могут изменяться

Применение:

таким образом процедура (и функция) может возвращать несколько значений

Запрещенные варианты вызова

```
Swap ( 2, 3 );       // числа
Swap ( x+z, y+2 ); // выражения
```

Программирование на языке Си

Тема 17. Функции

Функции

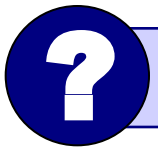
Функция – это вспомогательный алгоритм (подпрограмма), результатом работы которого является некоторое значение.

Примеры:

- вычисление модуля числа, $\sqrt{x}$
- расчет значений по сложным формулам
- ответ на вопрос (простое число или нет?)

Зачем?

- для выполнения одинаковых расчетов в различных местах программы
- для создания общедоступных библиотек функций



В чем отличие от процедур?

Функции

Задача: составить функцию, которая вычисляет наибольшее из двух значений, и привести пример ее использования

Функция:

тип результата

формальные параметры

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

return - вернуть результат функции

Функции

Особенности:

- в начале заголовка ставится **тип результата**

```
int  Max ( int a, int b )
```

- формальные параметры описываются так же, как и для процедур

```
float qq ( int a, float x, char c )
```

- можно использовать параметры-переменные

```
int Vasya (int & a, int & b )
```

- функции *обычно* располагаются до основной программы

Функции

Особенности:

- можно объявлять и использовать **локальные переменные**

```
float qq ( int a, int b)
{
  float x, y;
...
}
```

локальные переменные

Локальные переменные недоступны в основной программе и других процедурах и функциях.

Программа

```
int Max ( int a, int b )
{
  ...
}
main()
{
 int a, b, c;
 printf ( "Введите два числа\n" );
 scanf ( "%d%d", &a, &b );
 c = Max ( a, b );
 printf ( "Наибольшее число %d", c );
}
```

формальные параметры

фактические параметры

вызов функции

Задания

«4»: Составить функцию, которая определяет наибольший общий делитель двух натуральных и привести пример ее использования.

Пример:

```
Введите два числа:
14 21
НОД(14,21)=7
```

«5»: Составить функцию, которая вычисляет функцию синус как сумму ряда (с точностью 0.001)

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

***x* в радианах!**

Пример:

```
Введите угол в градусах:
45
sin(45) = 0.707
```

Логические функции

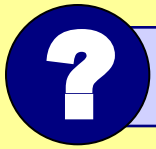
Задача: **составить функцию, которая определяет, верно ли, что заданное число – простое.**

Особенности:

- ответ – **логическое** значение: «да» (1) или «нет» (0)
- результат функции можно использовать как логическую величину **в условиях** (`if, while`)

Алгоритм: считаем число делителей в интервале от **2** до **N-1**, если оно не равно нулю – число составное.

```
count = 0;
for (i=2; i<N; i++)
  if (N % i == 0) count ++;
if ( count == 0 )
      // число N простое}
else // число N составное
```



Как улучшить?

Функция: простое число или нет

```
int Prime ( int N )
{
int count = 0, i;
for (i = 2; i*i <= N; i++)
  if (N % i == 0) count ++;
return (count == 0);
}
```

перебор только до $\sqrt{N}$

```
if (count == 0) return 1;
else            return 0;
```

Логические функции

```
#include <stdio.h>

int Prime ( int N )
{
...
}

main()
{
  int N;
  printf ( "Введите целое число\n" );
  scanf ( "%d", &N );
  if ( Prime( N ) )
       printf ("%d - простое число", N);
  else printf ("%d - составное число", N);
}
```

функция

Задания

«4»: Составить функцию, которая определяет, верно ли, что сумма его цифр – четное число.

Пример:

```
Введите число:          Введите число:
136                     245
Сумма цифр четная.      Сумма цифр нечетная.
```

«5»: Составить функцию, которая определяет, верно ли, что в заданном числе все цифры стоят по возрастанию.

Пример:

```
   Введите число:       Введите число:
258                     528
Верно.                  Неверно.
```

Основы программирования (на языке Си)

Тема 18. Рекурсия

Рекурсивные объекты

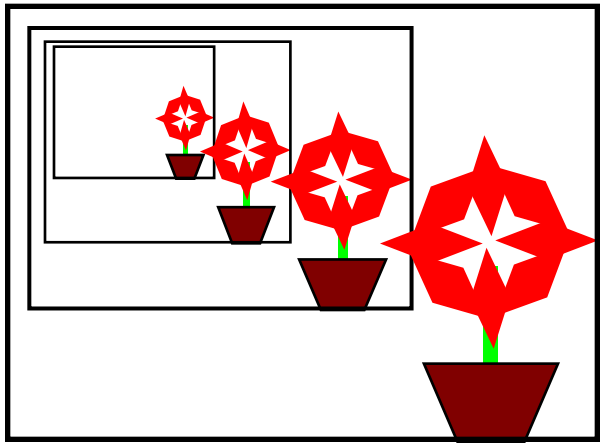
Примеры:

Сказка о попе и собаке:

У попа была собака, он ее любил.
Она съела кусок мяса, он ее убил.
В ямку закопал, надпись написал:

Сказка о попе и собаке

Рисунок с рекурсией:



Факториал:

$$N! = \begin{cases} 1, & \text{если } N = 1, \\ N \cdot (N-1)!, & \text{если } N > 1. \end{cases}$$

$$1! = 1, \quad 2! = 2 \cdot 1! = 2 \cdot 1, \quad 3! = 3 \cdot 2! = 3 \cdot 2 \cdot 1$$

$$4! = 4 \cdot 3! = 4 \cdot 3 \cdot 2 \cdot 1$$

$$N! = N \cdot (N-1) \cdot \boxtimes \cdot 2 \cdot 1$$

Рекурсивный объект – это объект, определяемый через один или несколько таких же объектов.

Рекурсия

Задача: составить рекурсивную функцию, которая вычисляет факториал числа, и привести пример ее использования

Функция:

тип результата

формальный параметр

```
int fact ( int a)
{
  if ( a==0 )
     return 1 ;
  else
     return a*fact(a-1);
}
```

return - вернуть результат функции

Рекурсивный вызов

Рекурсия

```
#include <stdio.h>

int fact ( int a )
{
...
}
main()
{
  int N;
  printf ( "Введите целое число\n" );
  scanf ( "%d", &N );
  printf ("Факториал %d равен %d",N,fact(N));
}
```

Рекурсивная функция

Задания

«5»: Составить рекурсивную функцию, которая вычисляет функцию синус как сумму ряда (с точностью 0.001)

Пример:

```
Введите угол в градусах:
45
sin(45) = 0.707
```

Основы программирования (на языке Си)

Тема 19. Массивы в процедурах и функциях

Массивы в процедурах

Задача: составить процедуру, которая переставляет элементы массива в обратном порядке.

параметр-массив

размер массива

```
void Reverse ( int A[] , int N )
{
int i, c;
for ( i = 0; i < N/2; i ++ ) {
  c = A[i];
  A[i] = A[N-1-i];
   A[N-1-i] = c;
  }
}
```

Массивы как параметры процедур

Особенности:

- при описании параметра-массива в заголовке функции его размер не указывается (функция работает с массивами **любого размера**)

Почему здесь размер не обязателен?

- размер массива надо передавать как отдельный параметр
- в процедура передается **адрес** исходного массива: все **изменения**, сделанные в процедуре **влияют** на массив в основной программе

Массивы в процедурах

```
void Reverse ( int A[], int N )
{
...
}
main()
{
  int A[10];
    // здесь надо заполнить массив
  Reverse ( A, 10 );  // весь массив
  // Reverse ( A, 5 );   // первая половина
  // Reverse ( A+5, 5 ); // вторая половина
}
```

это адрес начала массива в памяти

A или **&A[0]**

A+5 или **&A[5]**

Задания

«4»: Написать процедуру, которая сортирует массив по возрастанию, и показать пример ее использования.

«5»: Написать процедуру, которая ставит в начало массива все четные элементы, а конец – все нечетные.

Массивы в функциях

Задача: составить функцию, которая находит сумму элементов массива.

результат – целое число

параметр-массив

размер массива

```
int Sum ( int A[], int N )
{
int i, sum = 0;
for ( i = 0; i < N; i ++ )
   sum += A[i];
return sum;
}
```

Массивы в процедурах и функциях

```
int Sum ( int A[], int N )
{
...
}
main()
{
  int A[10], sum, sum1, sum2;
    // заполнить массив
  sum  = Sum ( A, 10 );  // весь массив
  sum1 = Sum ( A, 5 );   // первая половина
  sum2 = Sum ( A+5, 5 ); // вторая половина
  ...
}
```

Задания

«4»: Написать функцию, которая находит максимальный элемент в массиве.

«5»: Написать логическую функцию, которая определяет, верно ли, что среди элементов массива есть два одинаковых. Если ответ «да», функция возвращает 1; если ответ «нет», то 0.

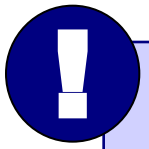
Подсказка: для отладки удобно использовать массив из 5 элементов, задаваемых вручную:

```
const int N = 5;
int A[N] = { 1, 2, 3, 3, 4 };
```

Основы программирования (на языке Си)

Тема 20. Строки в процедурах и функциях

Строки в процедурах и функциях



- строки передаются в функции и процедуры так же, как и массивы;
- функции и процедуры могут изменять строки – параметры.

Задача: составить процедуру, которая переставляет символы строки в обратном порядке.

Алгоритм:

- определить длину строки **len;**
- все символы первой половины переставить с соответствующими символами второй половины:

`s[i]` ⟷ `s[len-1-i]`

```
c = s[i];
s[i] = s[len-i-1];
s[len-1-i] = c;
```

Программа

```
void Reverse ( char s[] )
{
  int len = strlen(s);
  char c;
  for ( i = 0; i < len/2; i ++ ) {
    c = s[i];
    s[i] = s[len-i-1];
    s[len-1-i] = c;
    }
}
main()
{
  char s[] = "1234567890";
  Reverse ( s );
  puts ( s );
  Reverse ( s + 5 );
  puts ( s );
}
```

длину строки определяем на месте

Как сделать инверсию **любой** части строки?

```
0987654321
```

```
0987612345
```

Задания

«4»: **Разработать процедуру, которая переставляет пары соседних символов.**

Пример:

```
Введите предложение:
Вася пошел гулять!
Результат:
аВясп шолег лутя!ь
```

«5»: **Разработать процедуру, которая удаляет все лишние пробелы (в начале предложения и сдвоенные пробелы).**

Пример:

```
Введите предложение:
    Вася     пошел     гулять!
Результат:
Вася пошел гулять!
```

Символьные строки в функциях

Задача: составить функцию, которая находит количество цифр в строке.

```
int NumDigits ( char s[] )
{
  int i, count = 0;
  for ( i = 0; i < strlen(s); i ++ )
    if( strchr ( "0123456789", s[i] ) )
      count ++;
  return count;
}
```

```
if ( strchr ( "0123456789", s[i] ) != NULL )
```

или

```
if ( '0' <= s[i] && s[i] <= '9' )
```

Символьные строки в функциях

Основная программа

```
int NumDigits ( char s[] )
{
...
}
main()
{
  char s[80];
  int n;
  printf ( "Введите строку\n" );
  gets ( s );
  n = NumDigits ( s );
  printf ( "Нашли %d цифр.", s );
}
```

Задания

«4»: Разработать функцию, которая определяет, верно ли, что слово – палиндром.

Пример:

```
Введите слово:   Введите слово:
казак    кунак
Результат: Результат:
Это палиндром.   Не палиндром.
```

«5»: Разработать функцию, которая определяет, верно ли, что *предложение* (с пробелами) – палиндром.

Пример:

```
Введите предложение:
а роза упала на лапу азора
Результат:
Это палиндром.
```