Взаємодія фактичних і формальних параметрів функції.

При звертанні до функції всі її формальні параметри отримають значення відповідних фактичних параметрів. Кількість фактичних параметрів повинна строго відповідати кількості формальних. Самі ж фактичні параметри можуть бути довільними виразами, але значення яких має бути сумісним з типом відповідного формального параметра (сумісність означає, що значення виразу може бути перетворене до типу формального параметра). В тілі функції опрацьовуються копії значень фактичних параметрів, занесені в стек, а значення самих фактичних параметрів при цьому не змінюються.

Передача параметрів

Усі параметри, за винятком параметрів типу покажчик та масивів, передаються за значенням. Це означає, що при виклику функції їй передаються тільки значення змінних. Сама функція не в змозі змінити цих значень у викликаючій функції. Наступний *приклад* це демонструє:

```
# include <stdio.h>
 void test ( int a)
 {
    a=15;
    printf (" in test : a==%d\n", a);
 }
```

```
void main ()
 {
    int a =10;
    printf ("before test : a==%d\n", a);
    test (a);
    printf ("after test : a==%d\n", a);
 }
```

При передачі параметрів за значенням у функції утворюється локальна копія, що приводить до збільшення об'єму необхідної пам'яті. При виклику функції стек відводить пам'ять для локальних копій параметрів, а при виході з функції ця пам'ять звільняється. Цей спосіб використання пам'яті не тільки потребує додаткового її об'єму, але й віднімає додатковий час для зчитування.

Розглянемо функцію, яка визначає найбільше з трьох заданих дійсних значень:

```
double Max ( double x, double y , double z)
  {
    x = ( x > y ) ? x : y ;
    return ( x > z ) ? x : z ;
  }
```

У разі наступного оголошення змінних і звертання до функції

```
    double u, max ;
    max = Max ( u, 50, u*u/25 );
```

формальний параметр ***x*** функції отримає значення змінної ***u,*** параметр ***y*** – значення константи 50.0, а параметр ***z*** – значення виразу ***u*u/25*** **.** Всі три значення будуть перетворені до типу ***double***.

Такий взаємозв’язок параметрів називають звертанням за значенням *(call by value)*. У тілі функції параметр ***x*** змінюється, проте ці зміни не впливають на значення ***u*** **.**

Приклад функції, яка мала би міняти місцями значення двох аргументів:

```
void Swap1 ( int a, int b )
{
   int d = a ;
   a = b ;   b = d ;
}
```

Якщо цю функцію викликати наступним чином:

int al = 25 , bet = 18 ;

Swap1 (al, bet) ;

то значення ***al*** і ***bet*** не зміняться, хоча в тілі функції формальні параметри обмінюються значеннями (це не впливає на значення фактичних параметрів).

Щоб функція могла змінити значення змінної, треба передати у функцію адресу цієї змінної. Тоді з функції можна буде звертатись до змінної за адресою та оперувати безпосередньо з її значенням, а не з копією, занесеною в стек. Відповідний формальний параметр повинен бути вказівником, базовий тип якого збігається з типом змінної.

Таку організацію взаємозв’язку параметрів називають звертанням через посилання (*call by reference*).

Приклад – стандартна функція введення *scanf ( )*.

Запишемо правильний варіант функції обміну:

```
void Swap2 ( int *pa, int *pb )
  {
     int d = *pa ;
    *pa = *pb ;   *pb = d ;
  }
```

Звертання **Swap2 (&al, &bet) ;**

Масиви та символьні рядки як параметри функцій.

Якщо у функцію передається ім’я масиву, то на відміну від змінних інших типів не створюється копія масиву, а передається адреса його першого елемента. Тому функція здійснює роботу з тим масивом, який задано через фактичний параметр.

Використовуються дві рівнозначні форми оголошення формального параметру – вказівника на початок масиву

тип_елементів ім’я_масиву [ ] (наприклад ***int vect [ ]***)

тип_елементів * ім’я_вказівника (наприклад ***int * pv***)

Проілюструємо обидві форми прикладами функцій ***Average1( )*** та ***Average2( )***, кожна з яких обчислює середнє значення ***n*** послідовних елементів масиву дійсних чисел.

```
double Average1 ( double mas [ ], int n )
 {
   int i ;
   double sum = 0 ;
   for ( i = 0 ; i<n ; i++ )
      sum += mas [ i ] ;
   return sum / n ;
 }
```

```
double Average2 ( double *pa, int n )
 {
    int i ;
    double *pn, sum = 0 ;
    for ( pn = pa + n ; pa<pn ; pa++ )
       sum += *pa ;
    return sum / n ;
 }
```

Звертання до описаних вище функцій буде однаковим

double arr [150], ar1, ar2 ;

……..

ar1 = Average1 (arr, 150) ; */* ar1= Average2 (arr,150)*/*

ar2 = Average2 (&arr[10], 40) ;

Змінна ***ar1*** отримає середнє значення всіх 150 елементів масиву, змінна ***ar2*** – середнє значення елементів масиву, які мають індекси від 10 до 49.

Все сказане про параметри–масиви цілком стосується і тих параметрів функцій, які є символьними рядками. Їх можна оголошувати як масиви або як вказівники – остання форма на практиці використовується частіше.

Створення у функції нового масиву.

Значення, яке повертає функція, не може бути масивом чи символьним рядком. Щоб сформувати у функції новий масив чи символьний рядок, треба виділити місце для нього у викликаючій функції та включити до списку параметрів функції, яка формує масив (рядок), адресу виділеної для нього ділянки.

Інші варіанти – створення нового масиву (рядка) в динамічній пам’яті, або використання глобальних масивів і рядків (проте вони не захищені від випадкових змін, оскільки до них можна звертатись з кожної точки програми).

Приклад – функція, яка виділяє та повертає перше слово заданого речення. Аргументами функції є адреси фактичних параметрів. Така функція оперує безпосередньо з цими параметрами і може їх змінювати.

```
#include < stdio.h>
#include < ctype.h>
char * FirstWord ( char * s, char * word )
{
  char * w = word ;
  while ( isspase (* s ) )  s++ ;   /* пошук початку слова */
```

```
while ( *s !=‘ ‘ && *s !=‘,’ && *s !=‘ :’ *s !=‘ .’ *s !=‘ \0’)
            * w++ = *s++;     /* копіювання першого слова */
        * w = ‘\0’ ;
  return word ;
  }
 int main ( void )
 {
    char str [ ] = “ Наш факультет – найкращий” ;
char wrd [25] ;
  printf (“\n Перше слово: %s \n”, FirstWord (str, wrd )) ;
 return 0 ;
 }
```

Результат виконання

Перше слово: Наш

Щоб захистити значення фактичного параметра від випадкових змін у тілі функції, треба в оголошенні відповідного формального параметра вказати ***const*** . Якщо ж формальний параметр-вказівник оголосити ***const,*** то компілятор стежитиме, щоб у тілі функції значення цього вказівника не змінювалось.

Щоб зробити функцію ***FirstWord ( )*** захищеною, оголошення параметрів потрібно записати так:

char * FirstWord (const char * s, char * const word)