Imperatív programozás

Paraméterátadás

Kozsik Tamás és mások

ELTE Eötvös Loránd Tudományegyetem

Függvénydeklarációk és -definíciók C-ben

```c
int f( int n );
int g( int n ){ return n+1; }

int h();
int i(void);

int j(void){ return h(1); }

int h( int p, int q ){ return p+q; }

extern int k(int,int);

int printf( const char *format, ... );
```

Paraméterátadási technikák

- **Érték szerinti** (pass-by-value, call-by-value)
- Érték-eredmény szerinti (call-by-value-result) – Ada
- Eredmény szerinti (call-by-result) – Ada
- Cím szerinti (call-by-reference) – Pascal, C++
- **Megosztás szerinti** (call-by-sharing)
- Igény szerinti (call-by-need) – Haskell
- Név szerinti (call-by-name) – Scala
- Szövegszerű helyettesítés – C-makró

Érték szerinti paraméterátadás

```c
int lnko( int a, int b )
{
    int c;
    while( b != 0 ){
        c = a % b;
        a = b;
        b = c;
    }
    return a;
}
int main()
{
    int n = 1984, m = 365;
    int r = lnko(n,m);
    printf("%d %d %d\n",n,m,r);
}
```

```python
def lnko(a,b):
    while b != 0:
        a, b = b, a%b
    return a

n = 1984
m = 365
r = lnko(n,m)
print(n,m,r)
```

Bemenő szemantika

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

int main()
{
    int n = 1984, m = 365;
    swap(n,m);
    printf("%d %d\n",n,m);
}
```

```python
def swap(a,b):
    c = a
    a = b
    b = c

n = 1984
m = 365
swap(n,m)
print(n,m)
```

Mutató átadása érték szerint

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

```c
int main(){
   int *n, *m;
   n = (int*) malloc(sizeof(int));
   m = (int*) malloc(sizeof(int));
   if( n != NULL ){
      if( m != NULL ){
         *n = 1984; *m = 365;
         swap(n,m);
         printf("%d %d\n",*n,*m);
         free(n); free(m);
         return 0;      // success
      } else free(n);
   }
   return 1; // allocation failed
}
```

Cím szerinti paraméterátadás emulációja

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

```c
int main(){
    int n = 1984, m = 365;
    swap(&n,&m);
    printf("%d %d\n",n,m);
}
```

Cím szerinti paraméterátadás – Pascal

```
program swapping;

   procedure swap( var a, b: integer );  (* var: cím szerint *)
   var
      c: integer;
   begin
      c := a; a := b; b := c
   end;

   var n, m: integer;

begin
   n := 1984; m := 365;
   swap(n,m);
   writeln(n,' ',m)     (* 365 1984 *)
end.
```

Cím szerinti paraméterátadás – C++

```
#include <cstdio>

void swap( int &a, int &b )   /* &: cím szerint */
{
    int c = a;
    a = b;
    b = c;
}

int main()
{
    int n = 1984, m = 365;
    swap(n,m);
    printf("%d %d\n",n,m);
}
```

Érték-eredmény szerinti paraméterátadás: Ada

```
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;
procedure Swapping is

   procedure Swap( A, B: in out Integer ) is  -- be- és kimenő
      C: Integer := A;
   begin
      A := B; B := C;
   end Swap;

   N: Integer := 1984;
   M: Integer := 365;

begin
   Swap(N,M);
   Put(N); Put(M);  -- 365 1984
end Swapping;
```

Megosztás szerinti paraméterátadás

```c
void swap( int t[] )
{
    int c = t[0];
    t[0] = t[1];
    t[1] = c;
}
int main()
{
    int arr[] = {1,2};
    swap(arr);
    printf("%d %d\n",arr[0],arr[1]);
}
```

```python
def swap(t):
    t[0], t[1] = t[1], t[0]

arr = [1,2]
print(arr)
swap(arr)
print(arr)
```

Ez nem cím szerinti paraméterátadás

```c
void twoone( int t[] )
{
    int arr[] = {2,1};
    t = arr;
}

int main()
{
    int arr[] = {1,2};
    twoone(arr);
    printf("%d %d\n",arr[0],arr[1]);
}
```

```python
def twoone(t):
    t = [2,1]

arr = [1,2]
print(arr)
twoone(arr)
print(arr)
```

Automatikus változó visszaadása?

C: hibás

```c
int *twoone()
{
    int arr[] = {2,1};
    return arr;
}
```

Python: ok

Nem automatikus, hanem dinamikus tárolású változót ad vissza!

```python
def twoone():
    arr = [2,1]
    return arr

print(twoone())
```

Igény szerinti paraméterátadás

```
f True  a _ = a
f False _ b = b + b

main = print result
  where result = f False (fact 20) (fact 10)

        fact 0 = 1
        fact n = n * fact (n-1)
```

Szövegszerű helyettesítés

```
#define DOUBLE(n) 2*n
#define MAX(a,b) a>b?a:b

int main()
{
    printf("%d %d\n", MAX(10,100), DOUBLE(10));
    {
        int n = 5;
        printf("%d\n", DOUBLE(n+1));
        printf("%d\n", MAX(5,++n));
    }
}
```

Szövegszerű helyettesítés – becsapós

```c
#define DOUBLE(n) 2*n
#define MAX(a,b) a>b?a:b

int main()
{
    printf("%d %d\n", MAX(10,100), DOUBLE(10));
    {
        int n = 5;
        printf("%d\n", DOUBLE(n+1)); /* printf("%d\n", 2*n+1); */
        printf("%d\n", MAX(5,++n));
    }
}
```

Szövegszerű helyettesítés – zárójelezés

```c
#define DOUBLE(n) (2*(n))
#define MAX(a,b) ((a)>(b)?(a):(b))

int main()
{
    printf("%d %d\n", MAX(10,100), DOUBLE(10));
    {
        int n = 5;
        printf("%d\n", DOUBLE(n+1)); /* (2*((n)+1)) */
        printf("%d\n", MAX(5,++n));
    }
}
```

Szövegszerű helyettesítés – még így is veszélyes

```c
#define DOUBLE(n) (2*(n))
#define MAX(a,b) ((a)>(b)?(a):(b))

int main()
{
    printf("%d %d\n", MAX(10,100), DOUBLE(10));
    {
        int n = 5;
        printf("%d\n", DOUBLE(n+1)); /* (2*((n)+1)) */
        printf("%d\n", MAX(5,++n));  /* ((5)>(++n)?(5):(++n)) */
    }
}
```

Változó számú paraméter

```c
int printf( const char *format, ... );
```

```python
def sum( *args ):
    s = 0
    for n in args:
        s += n
    return s

sum()
sum(3)
sum(3,2)
sum(3,2,7,6,1,8)
```

Névvel jelölt paramétermegfeleltetés

```
def copy( src, dst ):
    for item in src:
        dst += [item]

a = [1,2,3]
b = [4,5]
copy( dst=b, src=a )
```

Paraméter alapértelmezett értéke

```
def unwords( words, separator=' ' ):
    length = len(words)
    if length == 0:
        return ''
    else:
        result = words[0]
        for i in range(1,length):
            result += separator
            result += words[i]
        return result

unwords(["alma","a","fa","alatt"])
unwords(["alma","a","fa","alatt"], separator='\n')
unwords(["alma","a","fa","alatt"], '\n')
unwords(separator='\n', words=["alma","a","fa","alatt"])
```

Változó számú paraméter után névvel jelölt paraméter(ek)

```
def unwords( *words, separator=' ' ):
    length = len(words)
    if length == 0:
        return ''
    else:
        result = words[0]
        for i in range(1,length):
            result += separator
            result += words[i]
        return result

unwords("alma","a","fa","alatt")
unwords("alma","a","fa","alatt", separator='\n')
```