Aplicatii tip MENU. Functii

BREVIAR TEORETIC

FUNCȚII

O funcție reprezintă o secvență de instrucțiuni ce poate fi apelată din mai multe puncte ale unui program, executând prelucrări determinate asupra unor date comunicate în momentul apelului.

FUNCȚII – DECLARARE, DEFINIRE, APEL

Putem împarți tipurile de date fundamentale ale limbajului C în tipuri de date numerice și tipul caracter. În mod normal nu exista diferențe funcționale între tipul caracter și tipul întreg, dar întâlnim o abordare diferita la modul de inițializare a unui caracter sau șir de caractere. Acest lucru se va observa în secțiunea dedicata tipului caracter, în cadrul laboratorului de față.

DEFINIREA FUNCȚIEI (CORPUL)

Corpul funcției va apărea în afara funcției main și în afara oricărei alte funcții din program. Dacă plasam acest corp de funcție (definiția funcției) după funcția main, vom fi nevoiți sa declaram funcția respectiva, declarație ce va apărea în fața primului apel al funcției.

Sintaxa generală:

```
[<tip_funcție> ] <id_funcție>([ <listă_declarații_parametri> ] )
{
     [ <declarații_identificatori_locali> ]
     [ <instrucțiuni> ]
     [ return [ <expresie> ]; ]
}
```

unde:

```
tip_funcție := int|char|unsigned|float|double|short|long
```

Exemplu:

```
//Corpul functiei - definitia
void putere (float b, unsigned char exp)
```

```
{
int i;
double rezultat = 1;
for( i=1; i<=exp; i++ )
     Rezultat *= b;
     printf ("\n%7.2f^%3d = %7.2lf", b, exp, rezultat);
}
```

DECLARAREA FUNCȚIEI

Declarația funcției este necesară compilatorului în momentul în care este nevoit să ”rezolve” apelul funcției. Dacă funcția a fost definită înaintea primului sau apel atunci declarația nu mai este necesară. În plus, în momentul declarării nu este necesară cunoașterea denumirii parametrilor formali ai funcției, așa că ei pot lipsi (ca nume, dar nu ca și tip).

Sintaxa generală:

```
[<tip_functie> ] <id_functie>([ <lista_declaratii_parametri> ] );
```

Exemplu:

```
//Declarea funcției
void putere( float, unsigned char );
```

APELUL FUNCȚIEI

Acesta poate fi utilizat în funcția main sau în oricare altă funcție din program, prin specificarea numelui funcției și, între paranteze rotunde, lista parametrilor reali, transmiși funcției. Parametri reali pot fi, în cazul general, expresii de tipul parametrilor formali asociați sau expresii de tipuri convertibile la acele tipuri.

Sintaxa generală:

```
<id_functie>( [ <lista_ parametri_reali> ] );
```

Exemplu:

```
void main()
{    int i;
     printf ("\nAfiseaza o serie de puteri succesive ale lui 2");
     for( i = 0; i < LIM; i++ )
          putere( 2, i );       //Apelul functiei
}
```

TRANSMITEREA PARAMETRILOR

Transmiterea valorilor parametrilor se face conform unor reguli similare celor de la inițializarea variabilelor.

Astfel, tipul fiecărui parametru actual este comparat cu cel al parametrului formal corespunzător și se realizează conversiile implicite de tip:

- *float* la *double*;
- *char* și *short* la *int*, etc.

Alte conversii necesare trebuie realizate prin operator cast.

Instrucțiunile din corpul funcției care se referă la parametri formali utilizează copiile valorilor parametrilor efectivi (valorile stocate de aceștia). Dacă se modifică valoarea unui parametru, de fapt este afectată doar copia, nu și parametrul efectiv corespunzător.

Exemplificare:

```
// Declarația funcției
double putere( double, int );

void main( void )
{
	int   p = 2;
	float f = 3.2f, val;

	// Apelul funcției
	val = putere( f-1, 3 );
	//... prelucrează "val"
}

// Corpul functiei
double putere( double b, int exp )
{
	double rez = 1;

	for ( ; exp; exp-- )
		rez *= b;
	return rez;
}
```

În cazul exemplului prezentat ”f-1, 3” reprezinta parametri reali, iar ”double b, int exp” sunt parametri formali.

În momentul apelului ”val = putere(f-1, 3);”, cele doua expresii sunt evaluate, iar rezultatele convertite la *double*, respectiv *int*.

La pasul următor valorile deja convertite vor fi COPIATE în parametrii “b”, respectiv “exp”.

PARAMETRI DE TIP ȘIR/MATRICE

În acest caz putem afirma că orice schimbare produsa în interiorul funcției va fi vizibilă și după revenirea din funcție. Acest lucru este posibil deoarece, în aceste cazuri, se face accesul la elementul unui șir pe baza adresei șirului, motiv pentru care accesul se va produce asupra șirului transmis ca parametru.

Exemplificare:

```
/*
 * Parametri de tip șir
 */
// Declararea unei funcții cu un parametru de tip ȘIR
void functiePSir( int n, int sir[] );
//void functiePSir( int n, int sir[ 100 ] );

int main( void )
{
```

```
     int N = 5, sirA[ 100 ];

      // Apelul funcției. Transmiterea unui șir ca parametru

     functiePSir( N, sirA );   //functiePSir( N, &sirA[0] );
     functiePSir( N/2, &sirA[ N/2 ] );

     return 0;
}
```

```
/*
 * Parametri de tip matrice
 */
// Declararea unei funcții cu un parametru de tip MATRICE
void functiePMat( int n, int m, int mat[][20] );
//void functiePMat( int n, int m, int mat[10][20]);

int main( void )
{
     int N = 5, M = 3,  matA[ 10 ][ 20 ];

     // Apelul functiei. Transmiterea unei matrici ca parametru
     functiePMat( N, M, matA );

     return 0;
}
```

APLICAȚII TIP MENU

Constau în crearea de programe capabile să afișeze o listă de opțiuni, etichetate prin diferite simboluri (litere, cifre, etc). Utilizatorul va putea alege, la un moment dat o etichetă, prin apăsarea unei taste, moment în care programul va executa o secvență de instrucțiuni specifice acelei opțiuni de meniu, urmând ca la finalul acestora să revină în modul de afișare a opțiunilor. Una dintre aceste opțiuni va fi de genul "X. Părăsire program.", opțiune ce va avea ca sarcina efectivă părăsirea aplicației.

SCHELETUL UNUI MENIU

Un model de aplicație tip *menu* este prezentat în continuare.

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

int main( void )
{    char c;
     do
     {
          //echivalent ca efect cu clrscr();
          system("cls");
          printf( "\n1. Optiunea 1 ..."
               "\n2. Optiunea 2 ..."
```

```
                //....
                "\n9. Informatii autor"
                "\n0. Exit"
                "\n------------------\n");

          switch( c = getch() )
          {
                case '0':
                     exit( 0 );
                     break;    // nu este obligatoriu... de ce?
                case '9':
                     printf( "\n[%c]\nAutor: RCP"
                          "\nSuceava - 2009"
                          "\n\n\t\t...press any key!", c );
                     getch();  //așteaptă o tastă
                     break;
                case '1':
                     printf( "\n[%c]...<empty>...", c );
                     getch();  //așteaptă o tastă
                     break;
                case '2':
                     printf( "\n[%c]...<empty>...", c );
                     getch();  //așteaptă o tastă
                     break;
                default :
                     printf( "\n[%c]...<NULL>...", c );
                     getch();  //așteaptă o tastă
                     break;    // nu este obligatoriu... de ce?
          }

     } while( !0 );           //buclă infinită

     // Nu este normal să ieșim pe aici. De ce?
     return -1;
}
```

MENIURI ȘI FUNCȚII

În cazul acestei abordări, opțiunea de menu va fi tratată prin intermediul apelului uneia sau mai multor funcții, create în cadrul programului. De exemplu, opțiunea '9' din menu va fi tratată astfel:

[Vechea abordare]

```
          case '9':
                printf( "\n[%c]\nAutor: RCP"
                     "\nSuceava - 2009"
                     "\n\n\t\t...press any key!", c );
                getch();       //așteaptă o tastă
                break;
```

[Noua abordare]

```
          case '9':
                fctDespre( c );
                break;
```

[... unde]

```
void fctDespre( char c )
{   printf( "\n[%c]\nAutor: RCP" "\nSuceava - 2009"
        "\n\n\t\t...press any key!", c );
    getch();     //a□teaptă o tastă
}
```

PROBLEME REZOLVATE

PROBLEME PROPUSE SPRE REZOLVARE

1. Scrieți un program ce va realiza diferite operații pe șiruri de numere întregi, și care să afișeze și sa implementeze opțiunile următorului meniu:
   C. Citire șir de numere.
   A. Afișare sir.
   S. Afișare sumă.
   F. Căutare element în vector.
   I. Info autor.
   X. Ieșire.
2. Scrieți un program care să afișeze și să implementeze opțiunile următorului meniu (se vor utiliza funcții):
   C. Citire matrice.
   A. Afișare matrice.
   S. Verificare dacă este simetrică sau nu.
   F. Căutare element în matrice și afișarea pozițiilor în care se găsește elementul (o poziție = o pereche de coordonate (x, y)).
   O. Afișare contur matrice.
   P. Parcurgerea în spirala a matricii se memorează într-un vector, apoi se afișează toate subșirurile de elemente consecutive. Aceasta opțiune va fi lansata în execuție numai dacă matricea este pătratică.
   F. Info autor.
   X. Ieșire.