Funcións

A colocación de información en memoria, sentencias de control, operacións aritméticas e operacións lóxicas constitúen a base da programación, e calquera aplicación se pode realizar exclusivamente en base a iso. Sen embargo, existen moitas accións de uso común nas aplicacións que necesitaríamos programar unha e outra vez para facer programas funcionais, e algunha delas requiren un coñecemento profundo do hardware para levalas a cabo (como por exemplo a visualización de información en pantalla ou a lectura de datos do teclado), é por iso que as distintas linguaxes de programación proporcionan unha ***libraría de funcións*** que incorpora unha serie de funcións de uso común na maior parte das aplicacións, como entrada por teclado, saída por pantalla, operacións matemáticas complexas (potencias, raíces, logaritmos ... etc), xestión de estruturas de datos.... etc.

Unha función e un conxunto de operacións para realizar unha operación concreta (como unha raíz cadrada ou un factorial) que se pode chamar mediante un identificador pasándolle a información necesaria para que realice o seu traballo (parámetros) e recuperar o resultado final das mesmas.

Exemplo (Uso de funcións da libraría da linguaxe):

```
<?php
        echo “Introduce un número: “;
        fscanf(STDIN, "%d\n", $numero); // lee numero de teclado
        $raiz = sqrt($numero); // calcula unha raíz cadrada
        echo “A raiz cadrada de $numero e $raiz”
?>
```

Tamén podemos definir as nosas propias funcións e facer uso de elas :

Exemplo:

```
<?php
        // Creamos a funcion factorial
        function factorial($numero) {
                $resultado=1;
                while($numero > 1)  {
                        $resultado = $resultado * $numero;
                        $numero = $numero – 1;
                }
                return $resultado;
        }
        // Programa
        echo “Introduce un número “;
        fscanf(STDIN, "%d\n", $numero); //  función que lee numero de teclado
        echo  “$numero ! = “ factorial($numero);  // función que calcula un factorial
?>
```

Creación de funcións

Unha función recibe información de entrada (ou non), a procesa mediante unha secuencia de instruccións e produce un resultado. As funcións reciben información nos seus ***parámetros***, que son variables dispoñibles para o corpo da función, e únicamente son accesibles dentro de esta, son locais á función. As variables creadas dentro da función son ***locais*** á función, e únicamente son accesibles dentro de ela, destruíndose cando remata a execución da función.

En xeral, unha variable sempre é local ao ámbito no que foi creada (bloque de código), sendo accesible nese bloque e nos bloques internos.

É moi importante comentar o significado dos parámetros recibidos e os valores devoltos.

Exemplo

/* **Función potencia**: Calcula a potencia dun número
Recibe: base, exp: valor da base e expoñente respectivamente
Devolve: $base^{exp}$

base e ***exp*** son a información que recibe a función para poder face-lo seu traballo. polo tanto, son os **parámetros**. Os valores que se lle pasen á función se copiarán nas variables parámetro correspondentes, que serán variables locais á función (so accesibles dentro da función). A función procesará os datos suministrados nos parámetros e voltará o resultado producido. Non convén que as funcións soliciten información (a deben recibir nos parámetros) nin que visualicen resultados (se deben retornar), xa que perderían a utilidade en casos xenéricos. Se debe solicitar a información, invocar a función cos valores para os parámetros, recoller o resultado e visualizalo */

```
función potencia(base,exp)
        cuenta=0
        signo=0
        Si (exp<0)
                signo=1
                exp=-exp
        Fin-Si
        resultado=1
        Mientras(cuenta<exponente)
                resultado=resultado*base
                cuenta=cuenta+1
        FinMientras
        Si (signo<0) resultado=1.0/resultado
        devolver resultado
fin-funcion


/* Este programa fai uso da funcion desenvolta para solicitar dous números e visualizar o resultado de elevar o primeiro número ao segundo */
Pedir n1
Pedir exp   // Ainda que se chame  igual que o parámetro, é unha variable distinta. Os parámetros so existen dentro da función
// o valor almacenado en n1 se copiará no parámetro base, o valor almacenado en exp se copiará ao parámetro exp
resultado=potencia(n1,exp);
Visualizar n1 "Elevado a " exp "=" resultado
```

As variables utilizadas so existen dentro do bloque de código no que se utilizan por primeira vez, e nos seus bloques internos.

Paso de parámetros

O paso de parámetros é o feito de pasar información dende un bloque de código a unha función especificandoa entre paréntese detrás do nome (ver exemplo anterior).

Cando pasamos información á función a información colocada na chamada se copia nas direccións de memoria indicadas polas variables definidas na cabeceira da función. Estas variables so se poderán utilizar no interior a función, xa que se creará o espacio en memoria no momento da chamada, e se liberará ao finalizar a execución da función.

En determinados casos e necesario que a función poda xenerar máis dun valor. Sen embargo, as funcións so poden retornar un valor. Para solucionar esto, emprégase o seguinte "truco": en lugar de pasarlle á función no parámetro a información coa que se vai a traballar, se lle pode pasar a DIRECCIÓN dunha variable externa, onde queremos que a función deixe a información. Este modo de traballo se coñece como ***paso de parámetros por referencia***, xa que en lugar de pasar o valor co que queremos traballar (***paso de parámetros por valor***) pasamos unha referencia (dirección de memoria) do sitio da memoria onde se atopan ou queremos deixar os datos.

O xeito de indicar o modo de paso de parámetros varía segundo a linguaxe, en seudocódigo podemos indicalo nun comentario ou utilizar un & antes do nome do parámetro.

Exemplo:

```
/* funcion ecuacion2g: resolve ecuacións de 2 grao
        recibe:  a, b, c: coeficientes de x^2, x e termo independente
                 r1,r2: referencias das variables onde deixamos os resultados
        devolve: o número de solucións da ecuación (0,1, ou 2)
r1 e r2 son parámetros por referencia, polo que cando modifiquemos os seus valores estaremos modificando
realmente o valor almacenado nas variables utilizada na chamada (porque realmente r1 e r2 serán a dirección
desas variables, as referenciarán). En cambio, calqueira cambio dos valores de a, b ou c non afectarán a
ningunha variable externa á función. */
funcion ecuacion2g(a,b,c,&r1,&r2)
        numr=0
        Si ((a!=0)||(b!=0))
                Si (a==0)
                        r1=-c/b;
                        numr=1
                SeNon
                        radicando=b*b-4*a*c
                        Si (radicando > 0)
                                raiz=raizCadrada(radicando)
                                r1=(-b+raiz)/2*a;
                                r2=(-b-raiz)/2*a;
                                numr=2;
                        FinSi
                FinSi
        FinSi
        devolver numr
fin-Funcion

/* NOTA: Evidentemente necesitamos elaborar a función para calcular raíces cadradas si a linguaxe non
dispón dunha na súa libraría de funcións. Supoñemos que a linguaxe dispón dela..... */
```

```
/* Programa que pide os coeficientes dunha ecuacion de 2º grao e visualiza o resultado */
Visualizar "Coeficiente de x² ?: "
Pedir  a
Visualizar "Coeficiente de x?:"
Pedir  b
Visualizar "Termo independente?: "
Pedir c
// a, b e c son variables DISTINTAS dos parámetroa a,b e c da función. Os seus valores se copiarán. Paso por valor.
// As direccións res1 e res2 se "asignarán" as variables r1 e r2 da función. Paso por referencia.
ok=ecuacion2g(a,b,c,res1,res2)
Si (ok == 0) Visualizar "Sen Solucións"
Se Non
        Si (ok == 1)
                Visualizar "Unha solución: " res1
        Se Non
                Visualizar "Dúas Solucións: " res1 " e " res 2
        Fin Si
Fin Si
```