INSTITUTO TECNOLÓGICO BELTRÁN

Centro de Tecnología e Innovación

SUBALGORITMOS (O SUBPROGRAMAS)

Cuando un programa comienza a ser largo y complejo (la mayoría de los problemas reales se solucionan con programas de este tipo) no es apropiado tener un único texto con sentencias una tras otra. La razón es que no se comprende bien qué hace el programa debido a que se intenta abarcar toda la solución a la vez. Asimismo el programa se vuelve monolítico y difícil de modificar. Además suelen aparecer trozos de código muy similares entre sí repetidos a lo largo de todo el programa.

Para solucionar estos problemas y proporcionar otras ventajas adicionales a la programación, los lenguajes de alto nivel suelen disponer de una herramienta que permite estructurar el programa principal como compuesto de subprogramas (rutinas) que resuelven problemas parciales del problema principal. A su vez, cada uno de estos subprogramas puede estar resuelto por otra conjunción de problemas parciales y así sucesivamente.

Los procedimientos y las funciones son mecanismos de estructuración que permiten ocultar los detalles de la solución de un problema y resolver una parte de dicho problema en otro lugar del código.

PROGRAMACIÓN TOP - DOWN

Top-down (de arriba hacia abajo) es una estrategia de procesamiento de información características de las ciencias de la información, especialmente en lo relativo al software. Por extensión se aplican también a otras ciencias sociales y exactas

En el modelo top-down se formula un resumen del sistema, sin especificar detalles. Cada parte del sistema se refina diseñando con mayor detalle. Cada parte nueva es entonces redefinida, cada vez con mayor detalle, hasta que la especificación completa es lo suficientemente detallada para validar el modelo.

El modelo top-down se diseña con frecuencia con la ayuda de cajas negras que hacen más fácil cumplir requisitos aunque estas cajas negras no expliquen en detalle los componentes individuales.

PROCEDIMIENTOS Y FUNCIONES

PROCEDIMIENTO

Un procedimiento es un fragmento de código cuya función es la de realizar una tarea específica independientemente del programa en el que se encuentre. Con los procedimientos se pueden crear algoritmos tales como la de modificación de datos, cálculos paralelos a la aplicación, activación de servicios, etc. Prácticamente cualquier cosa puede realizarse desde procedimientos independientes.

FUNCIÓN

Una función es exactamente lo mismo que un procedimiento salvo por un detalle: una función puede devolver un valor al programa principal y un procedimiento no. Con las funciones podemos realizar todas las tareas que se hacen con los procedimientos pero además pudiendo devolver valores (como por ejemplo el área de un triángulo).

Como se puede ver los conceptos de procedimiento y función son bastante similares, incluso podríamos definir un procedimiento como una función que no retorna ningún valor.

ESTRUCTURA Y USO

```
Tipo_Dato_Retorno Nombre (Parámetros) {
        sentencias;
        retorno; (sólo en el caso de las funciones)
}
```

Para un procedimiento el Tipo_Dato_Retorno es siempre void, o lo que es lo mismo, nada.

En el caso de las funciones se puede poner cualquier tipo de dato: int, float, char, etc.

Ahora vamos a plantear un algoritmo que calcule el área de un triángulo, veamos como se podría hacer mediante un procedimiento y una función.

PROCEDIMIENTO (EJEMPLO)

```
#include <stdio.h>

void pAreaTriangulo (void) {
   float base, altura;
   printf("Introduce base: ");
   scanf("%f",&base);
   printf("Introduce altura: ");
   scanf("%f",&altura);

   printf("El área es: %2.2f \n", (base*altura)/2);
}

int main()
{
 pAreaTriangulo();
 system("PAUSE");
}
```

FUNCIÓN (EJEMPLO)

```
#include <stdio.h>

float fAreaTriangulo (void) {
   float base, altura;
   printf("Introduce base: ");
   scanf("%f",&base);
   printf("Introduce altura: ");
   scanf("%f",&altura);

   return (base*altura)/2;
}

int main()
{

  float area;

  area = fAreaTriangulo();
  printf("El área es: %2.2f \n", area);

  system("PAUSE");
}
```

PROCEDIMIENTOS Y FUNCIONES

Recordemos: un procedimiento siempre se declarará con valor de retorno void y una función se declarará con cualquier tipo de dato como valor de retorno y en su código se añadirá la sentencia return y el valor que se desea devolver al programa que lo llama.

Es muy recomendable definir todos los procedimientos y funciones antes de la función principal main, el motivo de esto es que en C todas las funciones y procedimientos deben definirse antes de la primera línea de código en que se usa. En caso de que incumplamos ésta norma a la hora de compilar tendremos errores.

PROCEDIMIENTOS Y FUNCIONES

Para llamar a un procedimiento o función debemos hacerlo de la siguiente forma:

En un procedimiento escribimos su nombre seguido de los parámetros entre paréntesis:

```
pAreaTriangulo();
```

En una función debemos declarar antes una variable que almacene el dato que la función va a devolver, la variable debe ser del mismo tipo que el que retorna la función. Seguidamente llamamos a la función escribiendo la variable seguido del operador de asignación, el nombre de la función y los parámetros entre paréntesis:

```
float area;
area=areatriangulo();
```

Implementar funciones y procedimientos en nuestros programas es una muy buena práctica que se debe hacer siempre que se pueda, ayuda a mantener organizado el código y nos permite reutilizar funcionalidades en otras secciones del programa.

PARÁMETROS

A la hora de crear funciones y procedimientos puede que sea necesario que trabajemos con datos que se encuentran alojados en el programa que los llama, para eso existen los parámetros. Los parámetros son valores que se mandan a un procedimiento o función para que éstos puedan trabajar con ellos.

Para utilizar los parámetros en un procedimiento o función debemos declarar el tipo de dato y su identificador, exactamente igual que si declaramos una variable.

Es muy importante que las variables se pasen en el mismo orden que como se han definido en los parámetros. En caso de que no sea así podemos tener resultados inesperados en la ejecución del programa.

PARÁMETROS

Veamos el programa del área del triángulo pasando por parámetros los datos de base y altura:

```
#include <stdio.h>

void pAreaTriangulo (float base, float altura) {
   printf("El área es: %2.2f \n",(base*altura)/2);
}

int main()
{
 float base, altura;

 printf("Introduce base: ");
 scanf("%f",&base);
 printf("Introduce altura: ");
 scanf("%f",&altura);

 pAreaTriangulo(base,altura);

 system("PAUSE");
}
```

PASO DE PARÁMETROS POR VALOR Y POR REFERENCIA

El paso de parámetros a procedimientos y funciones es un recurso bastante potente de cara a trabajar con datos, además tenemos la posibilidad de pasar estos parámetros de varias formas, por valor o por referencia, esto lo especificamos cuando definimos el procedimiento o función.

El paso por valor es el que acabamos de ver, mediante el paso por valor el procedimiento o función reciben los valores indicados para poder trabajar con ellos, su declaración es la siguiente:

PROCEDIMIENTO

```
void pAreaTriangulo (float base, float altura)
```

FUNCIÓN

```
float fAreaTriangulo (float base, float altura)
```

A veces necesitaremos modificar varios de estos datos que hemos pasado en los parámetros, para ello es posible declarar los parámetros de tal forma que se puedan modificar, a esto se le llama paso por referencia.

PASO DE PARÁMETROS POR VALOR Y POR REFERENCIA

A veces necesitaremos modificar varios de estos datos que hemos pasado en los parámetros, para ello es posible declarar los parámetros de tal forma que se puedan modificar, a esto se le llama paso por referencia.

Gracias al paso por referencia un procedimiento o función no sólo realiza unas operaciones y devuelve (en el caso de las funciones) un valor, sino que también puede modificar los valores que se pasan como parámetros. Para poder realizar el paso de parámetros por referencia es necesario que en la declaración del procedimiento o función se especifique el espacio de memoria donde se encuentra alojado el dato, esto se realiza así:

PROCEDIMIENTO

```
void pAreaTriangulo (float *base, float *altura)
```

FUNCIÓN

```
float fAreaTriangulo (float *base, float *altura)
```

Lo que acabamos de definir como parámetros no son las variables en sí, sino un puntero que apunta a la dirección de memoria donde se encuentra alojado el valor, así podremos modificarlo.

PASO DE PARÁMETROS POR VALOR Y POR REFERENCIA

A veces necesitaremos modificar varios de estos datos que hemos pasado en los parámetros, para ello es posible declarar los parámetros de tal forma que se puedan modificar, a esto se le llama paso por referencia.

Gracias al paso por referencia un procedimiento o función no sólo realiza unas operaciones y devuelve (en el caso de las funciones) un valor, sino que también puede modificar los valores que se pasan como parámetros. Para poder realizar el paso de parámetros por referencia es necesario que en la declaración del procedimiento o función se especifique el espacio de memoria donde se encuentra alojado el dato, esto se realiza así:

PROCEDIMIENTO

```
void pAreaTriangulo (float *base, float *altura)
```

FUNCIÓN

```
float fAreaTriangulo (float *base, float *altura)
```

Lo que acabamos de definir como parámetros no son las variables en sí, sino un puntero que apunta a la dirección de memoria donde se encuentra alojado el valor, así podremos modificarlo.

PASO DE PARÁMETROS POR VALOR (EJEMPLO)

Veamos el caso de un programa que intercambia los valores de dos variables:

```
#include <stdio.h>

void pIntercambio(int num1, int num2){
 int aux;

 aux=num1;
 num1=num2;
 num2=aux;

 printf("El intercambio es: num1=%d
num2=%d\n",num1,num2);
}
```

```
int main()
{
  int num1=3, num2=5;

  printf("Vamos a intercambiar: num1=%d
num2=%d\n",num1,num2);

  pIntercambio(num1,num2);

  printf("El resultado tras el intercambio es:
num1=%d num2=%d\n",num1,num2);

  system("PAUSE");
}
```

PASO DE PARÁMETROS POR REFERENCIA (EJEMPLO)

Veamos el caso de un programa que intercambia los valores de dos variables:

```
#include <stdio.h>

void intercambio(int *num1, int *num2){
 int aux;

 aux=*num1;
 *num1=*num2;
 *num2=aux;

 printf("El intercambio es: num1=%d
num2=%d\n",*num1,*num2);
}
```

```
int main()
{
  int num1=3, num2=5;

  printf("Vamos a intercambiar: num1=%d
num2=%d\n",num1,num2);

  intercambio(&num1,&num2);

  printf("El resultado tras el intercambio es:
num1=%d num2=%d\n",num1,num2);

  system("PAUSE");
}
```

VARIABLES LOCALES Y GLOBALES

Gracias al concepto de procedimiento y función podemos tener dentro de un mismo programa varios subprogramas, los cuales se irán llamando entre ellos para poder realizar las tareas pertinentes, esto está muy bien pero es posible que necesitemos realizar tareas con variables alojadas en otros sitios, o necesitemos declarar nuevas, o simplemente no deseemos usar parámetros.

Para solucionar esto existe el concepto de variables locales y variables globales.

VARIABLES LOCALES

Son aquellas que declaramos al principio del cuerpo de un procedimiento o función, esto incluye a las declaradas en la función principal main.

VARIABLES GLOBALES

Son aquellas que las podemos definir en la zona de importación de librerías #include y la definición de constantes con #define. Su utilidad radica en poder operar con su valor en cualquier procedimiento o función sin necesidad de pasarla como parámetro.

VARIABLES GLOBALES (EJEMPLO)

Veamos el ejemplo del triángulo usando variables globales:

```
#include <stdio.h>

float base, altura;

void areatriangulo (void) {
   printf("El área es: %2.2f
\n",(base*altura)/2);
}

int main()
{
  printf("Introduce base: ");
  scanf("%f",&base);
  printf("Introduce altura: ");
  scanf("%f",&altura);

  areatriangulo();

  system("PAUSE");
}
```

VARIABLES GLOBALES (EJEMPLO)

Como se puede comprobar, el programa funciona exactamente igual que en su versión con parámetros, no obstante muchos desarrolladores no recomiendan hacer uso de ésta técnica debido a que en proyectos medios o grandes puede resultar confuso manejar variables globales sin un determinado estándar que regule su uso.

El modo más común de operar es mediante el uso de parámetros así que durante el curso y posteriormente seguiremos con esa técnica, no obstante si puede interesarte practicar los ejercicios que hemos visto hasta ahora usando variables globales.

RECURSIVIDAD

La recursividad no es más que la técnica de desarrollar una función que se llama a sí misma. Normalmente la veremos en funciones que realicen cálculos matemáticos, donde para realizar un cálculo repetitivo la función tendrá que llamarse a sí misma y así ir desarrollando el cálculo.

El ejemplo más típico que se suele utilizar para explicar la recursividad es el de un programa que calcula el factorial de un número, el factorial de un número responde a la siguiente fórmula matemática:

$$n! = n * (n-1)!$$

Es decir, que el factorial de un número es la multiplicación de ese número por el factorial del número inmediatamente anterior. Así si queremos saber el factorial de 5 sería:

$$5! = 5 * 4!$$

O lo que es lo mismo:

$$5! = 5 * 4 * 3 * 2 * 1$$

RECURSIVIDAD (EJEMPLO)

Veamos esto primero mediante un programa normal usando un bucle while:

```
#include <stdio.h>

int main()
{
  int num, num2;

  printf("Introduce número: ");
  scanf("%d",&num);
  num2=num;

  while (num2!=1){
        num2=num2-1;
        num=num*num2;
  }

  printf("El factorial es: %d \n", num);

  system("PAUSE");
}
```

RECURSIVIDAD (EJEMPLO)

Ahora veámoslo mediante el uso de una función recursiva:

```
#include <stdio.h>

int fFactorial(int num){
 if (num==0)
 {
   return 1;
 }
 else
 {
   return num * factorial(num-1);
 }
}
```

```
int main()
{
  int num;

  printf("Introduce número: ");
  scanf("%d",&num);
  printf("El factorial es: %d \n", fFactorial(num));

  system("PAUSE");
}
```

El resultado es el mismo pero el código ha variado bastante, la función recursiva ha cumplido perfectamente su cometido.

EJERCICIOS

1. Realice una función que determine el resultado de un cálculo de potencia

2. Escribir un programa que, utilizando funciones con parámetros, lea desde el teclado las unidades y el precio que quiere comprar, y en función de las unidades introducidas le haga un descuento o no (cuando las unidades excedan media docena se aplicará 4% y el 10% cuando excedan la docena)

3. Escriba un procedimiento que reciba un número del 1 al 10 y muestre por pantalla el número escrito en números romanos

4. Escriba un procedimiento que reciba dos valores numéricos y retorne mediante sus parámetros, el resultado de la multiplicación y la división entre ambos valores

5. Escriba una función que al recibir los valores de cantidad y monto; calcule el incremento del IVA (21%) sobre el total.

INSTITUTO TECNOLÓGICO BELTRÁN

Centro de Tecnología e Innovación