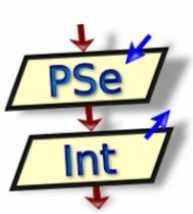
CURSO DE PROGRAMACIÓN FULL STACK

SUBPROGRAMAS CON PSEINT



EGG

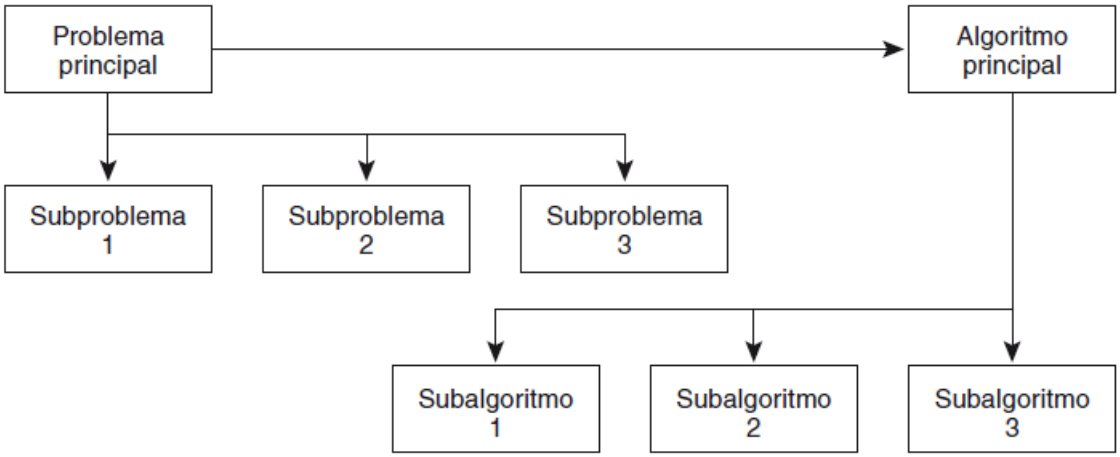
GUÍA DE SUBPROGRAMAS

SUBPROGRAMAS

Un método muy útil para solucionar un problema complejo es dividirlo en subproblemas —problemas más sencillos— y a continuación dividir estos subproblemas en otros más simples, hasta que los problemas más pequeños sean fáciles de resolver. Esta técnica de dividir el problema principal en subproblemas se suele denominar "divide y vencerás".

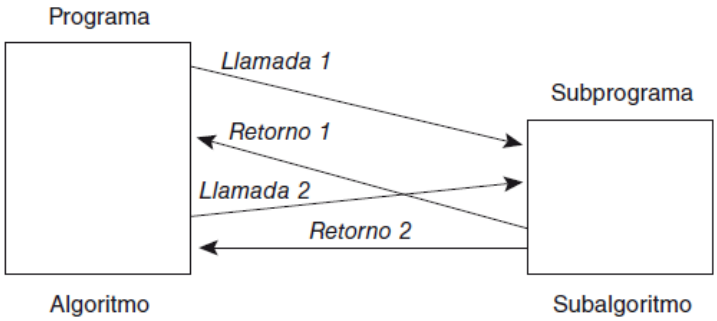
Este método de diseñar la solución de un problema principal obteniendo las soluciones de sus subproblemas se conoce como diseño descendente (top-down). Se denomina descendente, ya que se inicia en la parte superior con un problema general y el diseño específico de las soluciones de los subproblemas. Luego, las partes en que se divide un programa deben poder desarrollarse independientemente entre sí.

Las soluciones de un diseño descendente pueden implementarse fácilmente en lenguajes de programación y se los denomina subprogramas o sub-algoritmos si se emplean desde el concepto algorítmico.

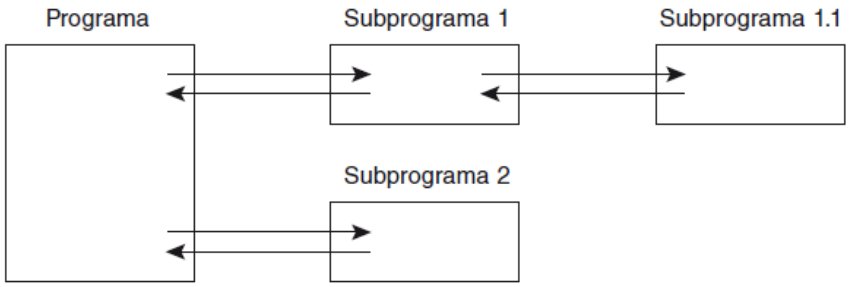


El problema principal se soluciona por el correspondiente programa o algoritmo principal y la solución de los subproblemas mediante subprogramas, conocidos como *procedimientos* o funciones. Un subprograma recibe *datos* desde el programa y le devuelve *resultados*.

Cada vez que el subprograma es llamado, el control retorna al lugar desde donde fue hecha la llamada, y a su vez, un subprograma puede llamar a otros subprogramas.



Un programa con un subprograma: función y procedimiento



Un programa con diferentes niveles de subprogramas

FUNCIONES

Matemáticamente una función es una operación que toma uno o más valores llamados *argumentos* y produce un valor denominado resultado (valor de la función para los argumentos dados).

Cada función se evoca utilizando su nombre en una expresión con los argumentos actuales o reales encerrados entre paréntesis. A una función no se le llama explícitamente, sino que se le invoca o referencia mediante un nombre y una lista de parámetros actuales.

El algoritmo o programa llama o invoca a la función con el nombre de esta última en una expresión seguida de una lista de argumentos que deben coincidir en cantidad, tipo y orden con los de la función que fue definida. La función devuelve un *único valor*.

DECLARACIÓN DE FUNCIONES

La declaración de una función requiere una serie de pasos que la definen. Una función como tal subalgoritmo o subprograma tiene una constitución similar a los algoritmos, por consiguiente, constará de una cabecera que comenzará con el tipo del valor devuelto por la función, seguido de la palabra función y del nombre y argumentos de dicha función. A continuación, irá el cuerpo de la función, que será una serie de acciones o instrucciones cuya ejecución hará que se asigne un valor al nombre de la función. Esto determina el valor particular del resultado que ha de devolverse al programa llamador.

SINTAXIS

```
Funcion variable_de_retorno <- Nombre (lista de parámetros formales)
      Definir variable_de_retorno como Tipo de Dato
      <acciones>   //cuerpo de la función
Fin Funcion
```

- Argumentos (parámetros formales); uno o más argumentos de la siguiente forma: (parámetro 1 [Por Valor/Por Referencia], [, parámetro 2 [Por Valor/Por Referencia],...]).

- Nombre asociado con la función, que será un nombre de identificador válido.

- <acciones> instrucciones que constituyen la definición de la función y que debe contener alguna instrucción mediante la cual se asigne un valor a la variable_de_retorno.

- variable_de_retorno contiene el resultado que devuelve la función que debe coincidir con el tipo de dato de retorno.

INVOCACIÓN A LAS FUNCIONES

Una función puede ser llamada de la siguiente forma:

```
nombre_función (lista de parámetros actuales)
```

- nombre_función*: función que va a llamar*

- lista de parametros actuales*: constantes, variables, expresiones.*

Cada vez que se llama a una función desde el algoritmo principal se establece automáticamente una correspondencia entre los parámetros formales y los parámetros actuales. Debe haber exactamente el mismo número de parámetros actuales que de parámetros formales en la declaración de la función y se presupone una correspondencia uno a uno de izquierda a derecha entre los parámetros formales y los actuales.

Una llamada a la función implica los siguientes pasos:

1. A cada parámetro formal se le asigna el valor real de su correspondiente parámetro actual.
2. Se ejecuta el cuerpo de acciones de la función.
3. Se devuelve el valor de la función y se retorna al punto de llamada.

PROCEDIMIENTOS

Aunque las funciones son herramientas de programación muy útiles para la resolución de problemas, con frecuencia se requieren subprogramas que calculen varios resultados en vez de uno solo, o que realicen la ordenación de una serie de números, etc. En estas situaciones la función no es apropiada y se necesita disponer del otro tipo de subprograma: el procedimiento.

Un procedimiento es un subprograma que ejecuta un proceso específico. Ningún valor está asociado con el nombre del procedimiento; por consiguiente, no puede ocurrir en una expresión. Un procedimiento se llama escribiendo su nombre. Cuando se invoca el procedimiento, los pasos que lo definen se ejecutan y a continuación se devuelve el control al programa que le llamó.

SINTAXIS

```
SubProceso Nombre (lista de parámetros formales)
      <acciones>
FinSubProceso
```

Los parámetros formales tienen el mismo significado que en las funciones.

INVOCACIÓN A UN SUBPROGRAMA

Un procedimiento puede ser llamado de la siguiente forma:

```
nombre [(lista de parámetros actuales)]
```

- nombre_procedimiento*: procedimiento que se va a llamar*
- lista de parametros actuales*: constantes, variables, expresiones.*

La lista de parámetros, formales en el procedimiento o actuales (reales) en la llamada se conoce como lista de parámetros.

PROCEDIMIENTO VERSUS FUNCIÓN

Los procedimientos y funciones son subprogramas cuyo diseño y misión son similares; sin embargo, existen unas diferencias esenciales entre ellos.

1. *Un procedimiento es llamado desde el algoritmo o programa principal mediante su nombre y una lista de parámetros actuales. Al llamar al procedimiento se detiene momentáneamente el programa que se estuviera realizando y el control pasa al procedimiento llamado. Después que las acciones del procedimiento se ejecutan, se regresa a la acción inmediatamente siguiente a la que se llamó.*

2. *Las funciones devuelven un valor, los procedimientos pueden devolver 0,1 o n valores y en forma de lista de parámetros.*
3. *El procedimiento se declara igual que la función, pero su nombre no está asociado a ninguno de los resultados que obtiene.*

ÁMBITO: VARIABLES LOCALES Y GLOBALES

Las variables utilizadas en los programas principales y subprogramas se clasifican en dos tipos: *variables locales y variables globales*.

Una ***variable local*** es aquella que está declarada y definida dentro de un subprograma, en el sentido de que está dentro de ese subprograma y es distinta de las variables con el mismo nombre declaradas en cualquier parte del programa principal. El significado de una variable se confina al procedimiento en el que está declarada. Cuando otro subprograma utiliza el mismo nombre se refiere a una posición diferente en memoria. Se dice que tales variables son locales al subprograma en el que están declaradas.

Una ***variable global*** es aquella que está declarada para el programa o algoritmo principal, del que dependen todos los subprogramas. La parte del programa/algoritmo en que una variable se define se conoce como ámbito o alcance (scope, en inglés).

El uso de variables locales tiene muchas ventajas. En particular, hace a los subprogramas independientes, con la comunicación entre el programa principal y los subprogramas manipulados estructuralmente a través de la lista de parámetros. Para utilizar un procedimiento sólo necesitamos conocer lo que hace y no tenemos que estar preocupados por su diseño, es decir, cómo están programados.

Una variable local a un subprograma no tiene ningún significado en otros subprogramas. Si un subprograma asigna un valor a una de sus variables locales, este valor no es accesible a otros programas, es decir, no pueden utilizar este valor. A veces, también es necesario que una variable tenga el mismo nombre en diferentes subprogramas. Por el contrario, las variables globales tienen la ventaja de compartir información de diferentes subprogramas sin una correspondiente entrada en la lista de parámetros.

COMUNICACIÓN CON SUBPROGRAMAS: PASO DE PARÁMETROS

Cuando un programa llama a un subprograma, la información se comunica a través de la lista de parámetros y se establece una correspondencia automática entre los parámetros formales y actuales. Los parámetros actuales son "sustituidos" o "utilizados" en lugar de los parámetros formales.

La declaración del subprograma se hace con

```
Subproceso nombre (F1, F2,…, Fn)
      <acciones>
FinSubproceso
```

y la llamada al subprograma con:

```
nombre (A1, A2, ..., An)
```

donde F1, F2, …, Fn son los parámetros formales y A1, A2, …, An los parámetros actuales o reales.

Los métodos más empleados para realizar el paso de parámetros son:

Paso por Valor

Los parámetros se tratan como variables locales y los valores iniciales se proporcionan copiando los valores de los correspondientes argumentos. Los parámetros formales (locales a la función o procedimiento) reciben como valores iniciales los valores de los parámetros actuales y con ello se ejecutan las acciones descritas en el subprograma.

Aunque el paso por valor es sencillo, tiene una limitación acusada: no existe ninguna otra conexión con los parámetros actuales, y entonces los cambios que se produzcan por efecto del subprograma no producen cambios en los argumentos originales y, por consiguiente, no se pueden pasar valores de retorno al punto de llamada: es decir, todos los parámetros son sólo de entrada. El parámetro actual no puede modificarse por el subprograma. La llamada por valor no devuelve información al programa que llama.

En PSeInt todas las variables escalares que hemos visto hasta el momento pasan por defecto "Por Valor" sino se especifica lo contrario explícitamente.

Paso por Referencia

En numerosas ocasiones se requiere que ciertos parámetros sirvan como parámetros de salida, es decir, se devuelvan los resultados al programa que llama. Este método se denomina paso por referencia o también de llamada por dirección o variable. El programa que llama pasa al subprograma la dirección del parámetro actual (que está en el ámbito del programa que llama). Una referencia al correspondiente parámetro formal se trata como una referencia a la posición de memoria, cuya dirección se ha pasado. Entonces una variable pasada como parámetro real es compartida, es decir, se puede modificar directamente por el subprograma.

La característica de este método se debe a su simplicidad y su analogía directa con la idea de que las variables tienen una posición de memoria asignada desde la cual se pueden obtener o actualizar sus valores. El área de almacenamiento (direcciones de memoria) se utiliza para pasar información de entrada y/o salida; en ambas direcciones.
En este método los parámetros son de entrada/salida y los parámetros se denominan parámetros variables.

RECURSIÓN

Una función o procedimiento que se puede llamar a sí mismo se llama recursivo. La recursión (recursividad) es una herramienta muy potente en algunas aplicaciones, sobre todo de cálculo. La recursión puede ser utilizada como una alternativa a la repetición o estructura repetitiva. El uso de la recursión es particularmente idóneo para la solución de aquellos problemas que pueden definirse de modo natural en términos recursivos.
La escritura de un procedimiento o función recursiva es similar a sus homónimos no recursivos; sin embargo, para evitar que la recursión continúe indefinidamente es preciso incluir una condición de terminación.

PREGUNTAS DE APRENDIZAJE

1. **Un subprograma es:**
   a) Un código especial que se utiliza para resolver distintos tipos de problemas.
   b) Un método de solucionar un problema complejo dividiéndolo en subproblemas
   c) Un método que siempre debe retornar algún resultado
   d) Ninguna de las anteriores

2. **En el paso de argumentos por valor se pueden utilizar:**
   a) Sólo variables
   b) Sólo variables o expresiones
   c) Sólo constantes o expresiones
   d) Variables, constantes o expresiones

3. **En el paso de argumentos por referencia se pueden utilizar:**
   a) Sólo variables
   b) Variables, constantes o expresiones
   c) Sólo variables o expresiones
   d) Sólo constantes o expresiones

4. **Una constante puede pasarse como argumento a una función:**
   a) Sólo por valor
   b) Sólo por referencia
   c) Por valor y por referencia
   d) No puede pasarse

5. **Una variable puede pasarse como argumento a un subprograma:**
   a) Sólo por valor
   b) Sólo por referencia
   c) Por valor y por referencia
   d) No puede pasarse como argumento

6. **¿Qué características tienen los elementos locales?**
   a) Son visibles en su ámbito y fuera
   b) Son invisibles en su ámbito y fuera
   c) Son invisibles en su ámbito y visibles fuera
   d) Son visibles en su ámbito e invisibles fuera

7. **¿Qué características tienen los elementos globales?**
   a) Son visibles en su ámbito y fuera
   b) Son invisibles en su ámbito y fuera
   c) Son visibles en su ámbito e invisibles fuera
   d) Son invisibles en su ámbito y visibles fuera

8. **Una función de un programa siempre debe**
   a) Recibir al menos un parámetro
   b) Realizar la llamada a un subprograma
   c) Devolver un resultado
   d) Ninguna de las anteriores

9. **Un procedimiento puede**
   a) No devolver ningún resultado
   b) Devolver n resultados
   c) Devolver sólo un resultado
   d) Todas las anteriores

10. **Cuando se escribe una función PSeInt**
   a) Es necesario definir el tipo de dato de la variable de retorno
   b) No es necesario definir el tipo de dato de la variable de retorno
   c) Es indiferente si el tipo de dato de la variable de retorno se define o no
   d) Ninguna de las anteriores