Tema 5: Estructuras básicas de la Programación Imperativa

En este tema nos introduciremos al concepto de **secuencia, selección e iteración.**

1. **Programación estructurada.**

La programación estructurada es una metodología de programación que busca construir programas fácilmente comprensibles.

# Representación de la estructura de un programa

Las estructuras de los programas imperativos se representan mediante **diagramas de flujo.** Las acciones se representan mediante rectángulos, y las condiciones mediante rombos, las cuales equivalen a preguntas a las que se pueden responder con un “si” o un “no”. El **flujo de control** se representa mediante líneas o vías que van de un elemento a otro. Se reconocen tres estructuras básicas para descomponer una acción compuesta en otras mas simples.

* **Secuencia:** estructura más sencilla, para emplear en la descomposición se debe utilizar una secuencia de acciones o partes que se ejecutan de forma sucesiva.
* **Selección:** Consiste en ejecutar una acción u otra, dependiendo de una determinada condición que se analiza a la entrada de la estructura. Si da como resultado “true” realiza una determinada acción y si da un resultado “false” realiza una acción diferente.
* **Iteración:** se trata de la repetición de una acción mientras se cumpla una determinada condición. La mas general es aquella en la que la condición se analiza a la entrada de la estructura, antes de iniciar cada nueva acción. También puede denominarse **bucle.**

1. **Expresiones condicionales.**

Para poder utilizar las estructuras de **selección** e **iteración** se precisa expresar las condiciones con las que controlar ambas estructuras. Esto se realiza mediante la construcción de expresiones condicionales. Estas expresiones solo pueden dan como resultado **Sí(true)** o **No(false)**.

Una forma de construir una expresión condicional es mediante el uso de operadores de condición en expresiones aritméticas. Esto permite realizar comparaciones entre dos valores del mismo tipo.

|  |  |  |
| --- | --- | --- |
| **Comparación** | **Símbolo matemática** | **Operador C+-** |
| Mayor que | > | > |
| Mayor o igual que |  | >= |
| Menor que | < | < |
| Menor o igual que |  | <= |
| Igual a | == | == |
| Diferente a |  | =! |

Con los operadores de comparación solo es posible realizar una única comparación entre valores. Sin embargo, es bastante normal que las condiciones sean más complejas. Pueden constituirse condiciones que impliquen a mas de dos valores como condiciones compuestas de carias condiciones simples. Las condiciones compuestas se construyen como expresiones lógicas, cada término de la expresión lógica puede considerarse una expresión condicional simple. Denominemos a E1 como condición simple 1 y a E2 como condición simple 2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operación lógica** | **Símbolo matemático** | **Operador c+-** |  |
| Conjunción (E1 y E2) | **^** | **&&** | **E1 y E2 SON CIERTOS** |
| Disyunción (E1 y E2) |  | **||** | **E1 Ó E2 ES CIERTO** |
| Negación (no E1) | **¬** | **!** | **E1 NO ES CIERTO** |

Si no se utilizan paréntesis el orden de prioridad de evaluación en el lenguaje C+- es el siguiente:

|  |  |  |
| --- | --- | --- |
| 1 | Operadores unarios | **! + - (SIGNO)** |
| 2 | Operadores multiplicativos | **\* / %** |
| 3 | Operadores aditivos | **+ -** |
| 4 | Operadores de comparación | **> >= < <=** |
| 5 | Operadores de igualdad | **== !=** |
| 6 | Operadores de Conjunción | && |
| 7 | Operadores de Disyunción | || |

NOTA: No están permitidas las operaciones entre distintos tipos de datos.

1. **Estructuras básicas C+-**

**SENTECIA IF.**

**if (**Condición**) {**

**Acción A;**

**} else {**

**Acción B;**

**}**

Se ejecuta la acción A si la condición se cumple(si devuelve el valor booleano **true**), en caso contrario se ejecutará la opción B; El formato de la sentencia if para la selección en cascada es el siguiente:

If(Condición 1){

Accion A;

}Else if(condición 2){

Acción B;

}Else{

Acción C;

}

## SENTENCIA WHILE

En C+- la estructura de iteración se consigue mediante la sentencia WHILE

while (Condición){

acción a;

}

### SENTENCIA F

Existen muchas situaciones en las que las repeticiones del bucle se controlan mediante una variable que va contando las veces que se ejecuta. La cuenta puede ser en sentido creciente o decreciente. La condición de la iteración se limita a comprobar si se ha alcanzado el limite correspondiente al numero de repeticiones previstas. En C+- se dispone de la sentencia FOR.

for(int Identificador\_Indice = inicial; Indice <= Final; Indice ++){

Acción;

}

El símbolo (;) separa los distintos elementos de control de la sentencia.

La variable **índice** sirve de contador para controlar el numero de iteraciones a realizar, inicialmente el valor **índice** toma el valor inicial y se incrementa automáticamente en una unidad con cada nueva ejecución de acción. Dicha acción se ejecuta repetidamente hasta que la variable Índice alcanza el valor Final.

NOTA: también se pueden hacer bucles dentro de otros bucles.