

Estructuras de control

Las estructuras de control nos permiten alterar el flujo de ejecución de las sentencias que componen el cuerpo de una función. Sin ellas, sólo podríamos ejecutar las líneas de un programa una a una en orden lineal, lo que no nos permitiría computar cosas demasiado interesantes!

Ejecución condicional: **IF**

Nos permite decidir, a partir del resultado de evaluar una *expresión booleana*, si ejecutar o no un bloque determinado, u optar entre 2 bloques posibles.

Sintaxis:

```
if (<expresión booleana>)  
    <bloque a ejecutar cuando la expresión es verdadera>
```

else

```
    <bloque a ejecutar cuando la expresión es falsa>
```

La sentencia **else** es opcional, puede utilizarse o no. En el caso de no utilizarlo, cuando la expresión evaluada sea falsa la ejecución continuará con la sentencia inmediatamente posterior al **if**.

Ejemplo:

```
int max (int a, int b)  
{  
    int c;  
    if (a>b)  
        c = a;  
    else  
        c = b;  
    return c;  
}
```

Ciclo, la estructura **WHILE**

Nos permite repetir la ejecución de un bloque hasta tanto una condición booleana se vuelva falsa.

Sintaxis:

```
while ( <expresión booleana> )  
    <bloque>
```

Ejemplo:

```
int factorial (int n)  
{  
    int res = 1;  
    while ( n > 1 )  
    {  
        res = res * n;  
        n = n - 1;  
    }  
    return res;  
}
```

Expresiones multivaluadas: SWITCH

Nos permite decidir entre diferentes bloques a ejecutar según el valor de una expresión.

Sintaxis:

```
switch ( <expresión ordinal> )  
{  
    case <valor 1>: <bloque1>; break;  
    case <valor 2>: <bloque2>; break;  
    .  
    .  
    .  
    case <valor n>: <bloque n>; break;  
    default: <bloque>  
}
```

Es muy importante la inclusión de la sentencia *break* luego de cada bloque, ya que es la que indica que ha terminado la ejecución de las instrucciones correspondientes a esa guarda. La falta de la misma, provocará que se siga ejecutando el siguiente bloque hasta que encuentre un *break* o el fin del *switch* (representado por la llave que cierra).

La guarda *default* es opcional y se ejecuta cuando no ha dado positiva ninguna comparación anterior.

Índice de funciones

-A-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
abort	stdlib	stdlib.h	cstdlib
abs	stdlib	stdlib.h	cstdlib
acos	math	math.h	cmath
asctime	time	time.h	ctime
asin	math	math.h	cmath
atan	math	math.h	cmath
atan2	math	math.h	cmath
atexit	stdlib	stdlib.h	cstdlib
atof	stdlib	stdlib.h	cstdlib
atoi	stdlib	stdlib.h	cstdlib
atol	stdlib	stdlib.h	cstdlib

-B-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
bsearch	stdlib	stdlib.h	cstdlib

-C-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
calloc	stdlib	stdlib.h	cstdlib
ceil	math	math.h	cmath
clearerr	stdio	stdio.h	cstdio
clock	time	time.h	ctime
cos	math	math.h	cmath
cosh	math	math.h	cmath
ctime	time	time.h	ctime

-D-

Función	Librería	Fichero de	Fichero de
---------	----------	------------	------------

Autor Norkis Pérez P. UNEFA

		cabecera C	cabecera C++
difftime	time	time.h	ctime
div	stdlib	stdlib.h	cstdlib

-E-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
exit	stdlib	stdlib.h	cstdlib
exp	math	math.h	cmath

-F-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
fabs	math	math.h	cmath
fclose	stdio	stdio.h	cstdio
feof	stdio	stdio.h	cstdio
ferror	stdio	stdio.h	cstdio
fflush	stdio	stdio.h	cstdio
fgetc	stdio	stdio.h	cstdio
fgetpos	stdio	stdio.h	cstdio
fgets	stdio	stdio.h	cstdio
floor	math	math.h	cmath
fmod	math	math.h	cmath
fopen	stdio	stdio.h	cstdio
formato	stdio	stdio.h	cstdio
fprintf	stdio	stdio.h	cstdio
fputc	stdio	stdio.h	cstdio
fputs	stdio	stdio.h	cstdio
fread	stdio	stdio.h	cstdio
free	stdlib	stdlib.h	cstdlib
freopen	stdio	stdio.h	cstdio
frexp	math	math.h	cmath
fscanf	stdio	stdio.h	cstdio

Autor Norkis Pérez P. UNEFA

fseek	stdio	stdio.h	cstdio
fsetpos	stdio	stdio.h	cstdio
ftell	stdio	stdio.h	cstdio
fwrite	stdio	stdio.h	cstdio

-G-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
getc	stdio	stdio.h	cstdio
getchar	stdio	stdio.h	cstdio
getenv	stdlib	stdlib.h	cstdlib
gets	stdio	stdio.h	cstdio
gmtime	time	time.h	ctime

-L-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
labs	stdlib	stdlib.h	cstdlib
ldexp	math	math.h	cmath
ldiv	stdlib	stdlib.h	cstdlib
localeconv	locale	locale.h	locale
localtime	time	time.h	ctime
log	math	math.h	cmath
log10	math	math.h	cmath
longjmp	setjmp	setjmp.h	csetjmp

-M-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
malloc	stdlib	stdlib.h	cstdlib
mblen	stdlib	stdlib.h	cstdlib
mbstowes	stdlib	stdlib.h	cstdlib
mbtowc	stdlib	stdlib.h	cstdlib

Autor Norkis Pérez P. UNEFA

memchr	string	string.h	cstring
memcmp	string	string.h	cstring
memcpy	string	string.h	cstring
memmove	string	string.h	cstring
memset	string	string.h	cstring
mktime	time	time.h	ctime
modf	math	math.h	cmath

-P-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
perror	stdio	stdio.h	cstdio
pow	math	math.h	cmath
printf	stdio	stdio.h	cstdio
putc	stdio	stdio.h	cstdio
putchar	stdio	stdio.h	cstdio
puts	stdio	stdio.h	cstdio

-Q-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
qsort	stdlib	stdlib.h	cstdlib

-R-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
raise	signal	signal.h	csignal
rand	stdlib	stdlib.h	cstdlib
realloc	stdlib	stdlib.h	cstdlib
remove	stdio	stdio.h	cstdio
rename	stdio	stdio.h	cstdio
rewind	stdio	stdio.h	cstdio

-S-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
scanf	stdio	stdio.h	cstdio
setbuf	stdio	stdio.h	cstdio
setjmp	setjmp	setjmp.h	csetjmp
setlocale	locale	locale.h	clocale
setvbuf	stdio	stdio.h	cstdio
signal	signal	signal.h	csignal
sin	math	math.h	cmath
sinh	math	math.h	cmath
sprintf	stdio	stdio.h	cstdio
sqrt	math	math.h	cmath
srand	stdlib	stdlib.h	cstdlib
sscanf	stdio	stdio.h	cstdio
strcat	string	string.h	cstring
strchr	string	string.h	cstring
strcmp	string	string.h	cstring
strcoll	string	string.h	cstring
strcpy	string	string.h	cstring
strcspn	string	string.h	cstring
strerror	string	string.h	cstring
strftime	time	time.h	ctime
strlen	string	string.h	cstring
strncat	string	string.h	cstring
strncmp	string	string.h	cstring
strncpy	string	string.h	cstring
strpbrk	string	string.h	cstring
strrchr	string	string.h	cstring
strspn	string	string.h	cstring
strstr	string	string.h	cstring
strtod	stdlib	stdlib.h	cstdlib

Autor Norkis Pérez P. UNEFA

strtok	string	string.h	cstring
strtol	stdlib	stdlib.h	cstdlib
strtoul	stdlib	stdlib.h	cstdlib
strxfrm	string	string.h	cstring
system	stdlib	stdlib.h	cstdlib

-T-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
tan	math	math.h	cmath
tanh	math	math.h	cmath
time	time	time.h	ctime
tmpfile	stdio	stdio.h	cstdio
tmpnam	stdio	stdio.h	cstdio
tolower	ctype	ctype.h	cctype
toupper	ctype	ctype.h	cctype

-U-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
ungetc	stdio	stdio.h	cstdio

-V-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
vfprintf	stdio	stdio.h	cstdio
vprintf	stdio	stdio.h	cstdio
vsprintf	stdio	stdio.h	cstdio

-W-

Función	Librería	Fichero de cabecera C	Fichero de cabecera C++
wctomb	stdlib	stdlib.h	cstdlib

Arreglos

En clase ya conocimos algunos tipos básicos como por ejemplo los tipos *char*, *int* y *float*. El lenguaje C++ permite, además, construir estructuras más complejas a partir de estos tipos básicos.

Una de las construcciones que podemos definir son los **arreglos**.

Arreglo: Colección *ordenada* de elementos de un mismo tipo. Ordenada significa que cada elemento tiene una ubicación determinada dentro del arreglo y debemos conocerla para accederlo.

Sintaxis:

Definición de un arreglo:

```
<tipo> nombre_variable[longitud];
```

Con esto diremos que *nombre_variable* es un arreglo de **longitud** elementos del tipo <tipo>. Cabe destacar que **longitud** debe ser cualquier expresión entera **constante** mayor que cero.

Asignación de un arreglo:

```
nombre_variable[índice] = expresión del tipo <tipo>
```

Esta instrucción asigna el valor asociado de la expresión a la posición **índice** del arreglo *nombre_variable*. El índice debe ser una expresión del tipo entero en el rango [0, **longitud**-1]. Cabe destacar que C++ no chequea que el valor de la expresión sea menor a **longitud**, simplemente asigna el valor a esa posición de memoria como si formara parte del arreglo, pisando, de esta manera, otros datos que **no forman** parte del mismo, con lo que finalmente el programa no funciona correctamente.

Acceso al contenido de un arreglo:

nombre_variable[índice] es valor del tipo <tipo> que puede ser asignado a una variable, o pasado como parámetro, imprimirlo, etc. Aquí también vale la aclaración de que el índice debe estar dentro del rango de definición del arreglo, C++ no chequeará que esto sea cierto y devolverá lo contenido en la posición de memoria correspondiente a un arreglo de mayor longitud, el dato obtenido de esta manera es basura.

Ejemplo:

Autor Norkis Pérez P. UNEFA

```
int a[5]; // Definición de un arreglo de 5 posiciones

void main()
{
    int i;

    // Pedimos el ingreso de 5 números
    for(i=0; i<4; i++)    //No olvidar que los arreglos van de 0 a longitud-1
    {
        cout << "Ingrese el elemento Nro: << i << endl;
        cin >> a[i];
    }

    // Los imprimimos
    imprimir(a,5);
}

void imprimir(int b[], int tamaño)
{
    int i;

    for(i=0; i<tamaño; i++)
    {
        cout << "Nro: << i << " << b[i] << endl;
    }
}
```

Ahora veamos un ejemplo, donde se trata un arreglo de 10 elementos con ciclos `for`:

```
//Uso de arreglos en C++
#include <iostream>
using std::cout;
using std::cin;
```

Autor Norkis Pérez P. UNEFA

```
using std::endl;
int main()
{
    int arregloEntero[10] = {0};
    //Arreglo entero de 10 elementos inicializados todos en 0.
    cout << "Arreglo recién declarado: " << endl;
    for (int i = 0 ; i < 10 ; i++)
        //Notar el menor estricto (<) para ir desde 0 hasta 9
        cout << "arregloEntero["<<i<<"]="<<arregloEntero[i] << endl;
    cout << "Introduzca 10 nuevos valores " << endl;
    for (int i = 0 ; i < 10 ; i++)
        //Notar el menor estricto (<) para ir desde 0 hasta 9
        {
            cout << " Introduzca nuevo valor para
arregloEntero["<<i<<"]" << endl;
            cin >> arregloEntero[i];
        }
    cout << "Luego de los valores introducidos, el arreglo quedo asi: " << endl;
    for (int i = 0 ; i < 10 ; i++)
        //Notar el menor estricto (<) para ir desde 0 hasta 9
        cout << "arregloEntero["<<i<<"]="<<arregloEntero[i] << endl;
    return 0;
}
```