

Programación Básica
Estructuras de control: *for*, *while*, *do-while*

Iván Cabrera Altamirano
`ivan.cabrera@embedded.com.mx`

Enero 21, 2010

¿Qué es una estructura de control?

¿Qué es una estructura de control?

Definition

Son aquellas que permiten al programador alterar el flujo de ejecución de un programa.

Existen tres tipos de estructuras de control:
de secuencia, de selección, **de repetición**.

De acuerdo al **teorema del programa estructurado** (Bohm & Jacopini, 1966), cualquier programa puede escribirse utilizando únicamente esos 3 tipos de estructuras.

¿Donde puedo utilizar estructuras de control repetitivas?

¿Donde puedo utilizar estructuras de control repetitivas?

- Acceder un arreglo.

¿Donde puedo utilizar estructuras de control repetitivas?

- Acceder un arreglo.
- Procesar una cadena de caracteres.

¿Donde puedo utilizar estructuras de control repetitivas?

- Acceder un arreglo.
- Procesar una cadena de caracteres.
- Realizar operaciones matemáticas (e.g. promedio, integración numérica).

¿Donde puedo utilizar estructuras de control repetitivas?

- Acceder un arreglo.
- Procesar una cadena de caracteres.
- Realizar operaciones matemáticas (e.g. promedio, integración numérica).
- Ordenar y buscar datos.

¿Donde puedo utilizar estructuras de control repetitivas?

- Acceder un arreglo.
- Procesar una cadena de caracteres.
- Realizar operaciones matemáticas (e.g. promedio, integración numérica).
- Ordenar y buscar datos.
- **Ejecutar acciones más de una ocasión.**

Estructura de control repetitiva: *for*

Se utiliza para ejecutar acciones repetitivas con un número conocido de iteraciones.

```
for(inicialización; condición; incremento) {  
    expresiones a repetir;  
}
```

Ejemplo utilizando la estructura de control repetitiva *for*

```
for(i=0; i<10; i++) {  
    printf("Elemento [%d] = %d\n",i,valores[i]);  
}
```

- Inicialización: $i = 0$;
- Condición booleana de parada: $i < 10$;
- Incremento: $i + +$;
- Expresión a repetir: *printf(...)*;

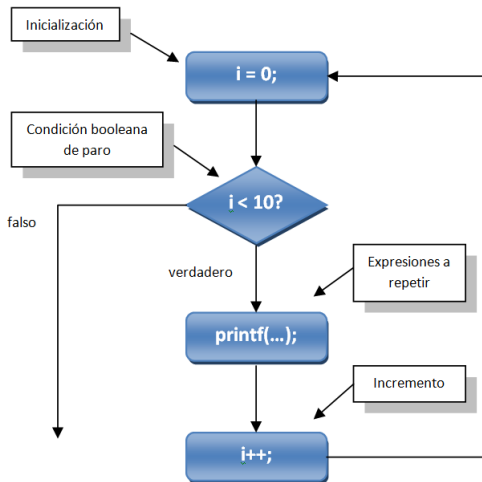


Figura: Diagrama de flujo del ciclo *for*

Estructura de control repetitiva: *while*

Se utiliza para ejecutar acciones repetitivas mientras se cumpla o no una condición. Es ideal cuando no sabemos previamente el número de iteraciones.

```
inicialización;
while(condición) {
    expresiones a repetir;
    incremento;
}
```

Ejemplo utilizando la estructura de control *while*

```
i = 0;
k = getchar();

while(k != '\0') {
    buffer[i] = k;
    k = getchar();
    i = i + 1;
}
buffer[i] = '\0';
```

- Inicialización: $i = 0$;
- Condición booleana de parada: $k \neq '\0'$;
- Expresión a repetir: $buffer[i] = k, k = getchar()$;
- Incremento: $i = i + 1$;

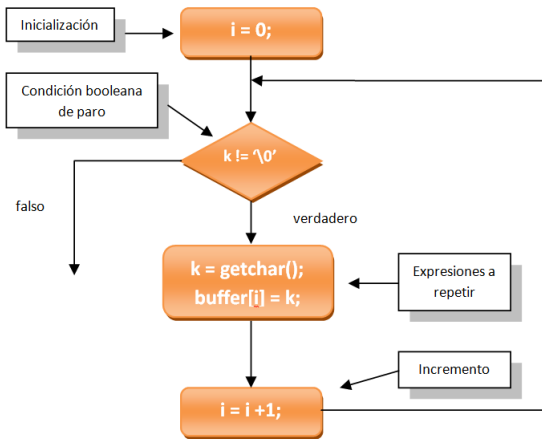


Figura: Diagrama de flujo del ciclo *while*

Estructura de control repetitiva: *do-while*

Se utiliza para ejecutar acciones repetitivas mientras se cumpla o no una condición. Es ideal cuando no sabemos previamente el número de iteraciones.

Esta estructura de control nos ofrece una ventaja adicional al ciclo *while*, ya que la sentencia se ejecutará por lo menos una vez.

```
inicialización;  
do {  
    expresión a repetir;  
    incremento;  
} while (condición booleana)
```

Ejemplo utilizando *do-while*

```
i = 0;
do {
    k = getchar();
    buffer[i] = k;
    i++;
} while(k != '\0')
```

- Inicialización: $i = 0$;
- Expresión a repetir: $buffer[i] = k, k = getchar()$;
- Incremento: $i + +$;
- Condición booleana de parada: $k != '\0'$;

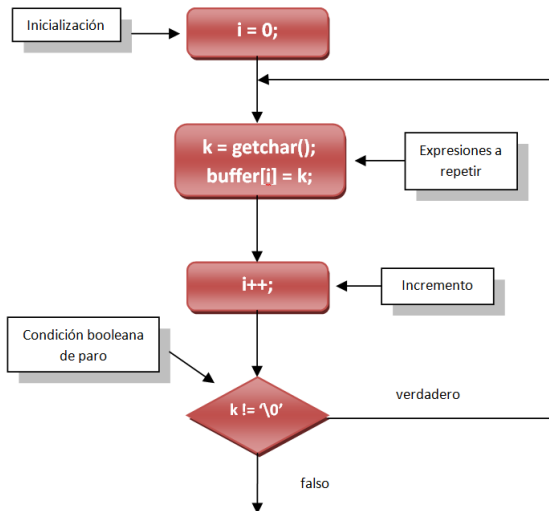


Figura: Diagrama de flujo del ciclo *do-while*

Algunos otros ejemplos

Algunos otros ejemplos

- Ocupando varias inicializaciones e incrementos.

```
for(i = 0, j = 0; i < 20; i = i + 2, j++) {  
    num16bits[j] = num8bits[i] + num8bits[i+1]*256;  
}
```

Algunos otros ejemplos

- Ocupando varias inicializaciones e incrementos.

```
for(i = 0, j = 0; i < 20; i = i + 2, j++) {  
    num16bits[j] = num8bits[i] + num8bits[i+1]*256;  
}
```

- Utilizando decrementos en lugar de incrementos.

```
signed int i;  
i = 20;  
printf("Imprimiendo en orden descendente\n");  
while(i >= 0) {  
    printf("buffer[%d] = %d\n", i, num8bits[i]);  
    i--;  
}
```

Algunos otros ejemplos

Algunos otros ejemplos

■ Ciclos anidados

```
for (i=0; i<5; i++){  
    for(j=0; j<5; j++) {  
        printf("datos[%d] [%d] = %d\n",i,j,datos[i][j]);  
    }  
}
```

Probables errores

Probables errores

- Inicialización incorrecta:

```
int acumulador = 0;
```

```
while(acumulador < 20000) {  
    acumulador = acumulador * 5;  
}
```

Probables errores

- Inicialización incorrecta:

```
int acumulador = 0;

while(acumulador < 20000) {
    acumulador = acumulador * 5;
}
```

- Condiciones mal escritas:

```
int contador = 1;

while(contador != 10) {
    contador += 2;
}
```

Probables errores cont.

Probables errores cont.

- Alcance de las variables:

```
unsigned char i;  
for (i = 0; i<10000; i++) {  
    printf("Contador: %d\n",i);  
}
```

Probables errores cont.

- Alcance de las variables:

```
unsigned char i;
for (i = 0; i<10000; i++) {
    printf("Contador: %d\n",i);
}
```

- Condiciones con numeros flotantes:

```
float contador = 0.0f;
while(contador != 1.0f) {
    contador = contador + 0.33333333f;
}

float contador = 0.0f;
while(contador != 1.0f) {
    contador = contador + 0.33333333f;
}
```