UNIVERSIDAD DE MURCIA

Introducción a la Programación

Grado en Ingeniería Informática

10. Iteraciones

(Composición iterativa)

Bloque 3. Iteración
10. Iteraciones

Dr. Isidro Verdú

Introducción: necesidad de iterar

Problema: calcular la media de un conjunto de enteros positivos que se introducen por teclado. Para terminar la instrucción se introduce un número negativo.

Necesitamos sumar todos los datos, y saber el número de datos. luego dividir suma/nDatos

```
Algoritmo
  suma   ← 0
  nDatos ← 0
  Leer (nota)
  SI nota ≥ 0
      suma   ← suma + nota
      nDatos ← nDatos + 1
      Leer (nota)
      SI nota  ≥ 0
         suma ← suma + nota
         nDatos ← nDatos + 1
         Leer (nota)
         SI ...
```

2

Introducción: necesidad de iterar

Problema: calcular la media de un conjunto de enteros positivos que se introducen por teclado. Para terminar la instrucción se introduce un número negativo.

Necesitamos iteraciones

```
Léxico
    Suma, nDatos, nota : entero;
    media: real;

Algoritmo
    Suma ← 0; Ndatos ← 0;
    Leer (nota);
    Mientras (nota>0)
        Suma ← Suma + nota;
        nDatos ← nDatos +1;
        Leer (nota);
    Fin_mientras
    media = suma / Ndatos;
    Escribir (media)
Fin.
```

3

Composición Iterativa MIENTRAS

Sintaxis: MIENTRAS <condición >
<sentencias>
FIN_MIENTRAS

Donde:
<condición> : expresión booleana. Condición de continuación. Al inicio
<sentencias> una secuencia de acciones de todo tipo

Si la primera vez no se cumple la condición no entra en el cuerpo de la iteración

Esquema general

Acciones de inicio
MIENTRAS condición
Acciones de la Iteración
FIN_MIENTRAS
Acciones finales

4

Composición Iterativa HACER - MIENTRAS

Sintaxis: HACER
<sentencias>
MIENTRAS <condición >

Donde:
<condición> : expresión booleana. Condición de continuación. Al inicio
<sentencias> una secuencia de acciones de todo tipo

Siempre entra la primera vez en el cuerpo de la iteración

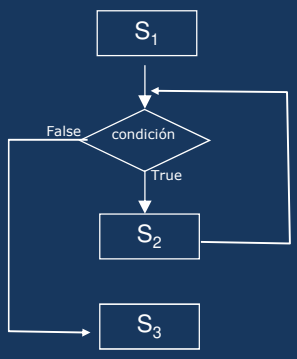
Esquema general

Acciones de inicio
HACER
Acciones de la Iteración
MIENTRAS condición
Acciones finales

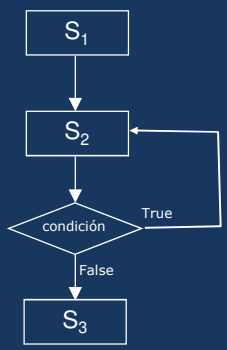
5

Composiciones Iterativas

Mientras



Hacer - Mientras



6

Ejemplo

Contar de n a 5

```
Léxico
    n: entero;
Algoritmo
    Leer(n)
    Mientras (n<=5)
        Escribir (n);
        n ← n+1;
    Fin_mientras
Fin.
```

n=3: 3,4,5
n=5: 5
n=6: -

```
Léxico
    n: entero;
Algoritmo
    Leer(n)
    Hacer
        Escribir (n);
        n ← n+1;
    Mientras (n<=5)
Fin.
```

n=3: 3,4,5
n=5: 5
n=6: 6

7

Iteraciones en C/C++

while

```
#include <stdio.h>
main() {
    int contador;

    contador=0;

    while (contador<10)
    {
        printf("Número %d\n",contador);
        contador = contador+1;
    }
}
```

while: Se repite el bloque mientras se cumpla la condición.

while es una palabra reservada.

8

Iteraciones en C/C++

```
while (condición)
   sentencia;
```

"sentencia;" puede ser una sola instrucción, o un bloque de instrucciones entre llaves

```
while (condición)
{
   sentencia;
   sentencia;
   …
}
```

9

Iteraciones en C/C++

```
contador=0;
while (contador<10)
{
   printf("Número %d\n",contador);
   contador=contador+1;
}
```

En bucles sencillos existe una *variable de control del bucle*.

La variable de control tiene que ser ***inicializada, comprobada y actualizada***.

La condicion puede ser cualquier expresión

```
a=b=x=2;
while ( (a*b)+2 != x+8)
{
   a=a+(c*2);
   x=x+1;
}
```

(ojo: este ejemplo no hace nada concreto)

10

do - while

```
do
   sentencia;
while (condición);
```

"sentencia;" puede ser una sola instrucción, o un bloque de instrucciones entre llaves

```
do
{
   sentencia;
   sentencia;
   …
}
while (condición);
```

11

while

```
scanf("%d",&a);

while (a<10)
{
      printf("a: %d\n",a);
      a++;
}
```

***while** evalúa la condición al principio; puede entrar al bucle o no.*

do while

```
scanf("%d",&a);

do
{
      printf("a: %d\n",a);
      a++;
} while (a<10);
```

***do while** evalúa la condición al final; siempre entra al menos una vez.*

12

estructura for

(para)

como while pero agrupa la inicialización, condición y actualización

13

Iteraciones

```
para (inicialización, condición, actualización)
    sentencias
fin_para
```

```
para (i ← 0, i<23, i ← i+1)
    v ← i * i * i
    Escribir ( v );
fin_para
```

14

Iteraciones en C/C++

```
contador=0;
while (contador < 5)
{
   printf("%d ",contador);
   contador = contador + 1;
}
```

Inicialización

Condición

Actualización

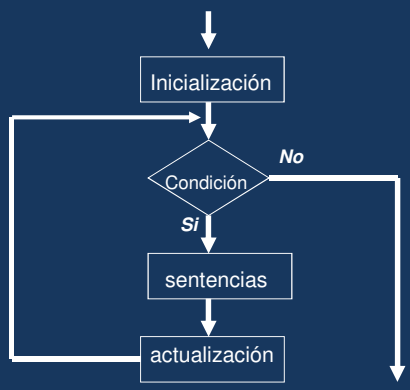
Inicialización *Condición* *Actualización*

```
for (contador = 0; contador < 5; contador ++)
{
    printf("%d ",contador);
}
```

15

Iteraciones en C/C++

for (*inicialización*; *condición*; *actualización*)
sentencia;



"sentencia;" puede ser una sola instrucción, o un bloque de instrucciones entre llaves

```
for (inicializac; condición; actualizac)
{
    sentencia;
    sentencia;
    …
}
```

16

Iteraciones en C/C++

```
multi=1;
sum=0;

for (i = 0; i < 5; i ++)
{
      multi=multi*2;
      sum=sum+2;
}

printf("%d %d",multi,sum);
```

for es muy útil para repetir un número conocido de veces

```
multi=1;
sum=0;
i=0;
while (i<5)
{
      multi=multi*2;
      sum=sum+2;
      i++;
}

printf("%d %d",multi,sum);
```

17

Iteraciones en C/C++

En C/C++ la inicialización, condición y actualización pueden ser expresiones complejas

```
for (a = 3*x+b; (a*b)+2 != x+8; a=c*2+sqrt(h*2) )
{
   …
   …
   …
}
```

18

Iteraciones en C/C++

```
double p;
for (p=0.75; p<=5.5; p+=0.25)
      sentencia;
```

```
double p;
for (p=pow(y,3.0); p>2.0; p=sqrt(p))
      sentencia;
```

Instrucción mucho más potente que en otros lenguajes

```
for ( i=0 ; i<5 ; printf("i: %d\n",i++) ) ;
```

¡no hay sentencias!

19

Bucles anidados

```
for ( x = 1 ; x <= ultimox ; x++ )
      for (y = 1 ; y <= ultimoy ; y++ )
      {
            producto=x*y;
            printf("%d x %d = %d\n",x,y,producto);
      }
```

```
for ( x = 1 ; x <= ultimox ; x++ )
{
      producto=x*y;
      for (y = 1 ; y <= ultimoy ; y++ )
            printf("%d, ",y);
      printf("\n%d\n",x);
}
```

(ojo: este ejemplo no hace nada concreto)

20

Integración en una programa general

```
for ( x = 1 ; x <= ultimox ; x++ )
{
      producto=x*y;
      z=5;
      while (z<100)
      {
        for (y = 1 ; y <= ultimoy ; y++ )
              printf("%d, ",y);
        h=sqrt(2+x*z)+media(a,b,c);
      }
      printf("\n%d\n",x);
}
```

(ojo: este ejemplo no hace nada concreto)

21

Centinelas y banderas

Bucles controlados por ***centinelas***

Se puede usar un valor especial (centinela) como valor de finalización

Este programa suma los números que se introducen

Acaba cuando introduzca el valor -1

```
char centinela=-1; int valor, suma=0;

printf ("Introduce valor (-1 para acabar):");
scanf("%d",&valor);

while (valor != centinela)
{
      suma=suma+valor;
      printf ("Introduce valor (-1 para acabar):");
      scanf("%d",&valor);
}
printf("Suma: %d\n",suma);
```

22

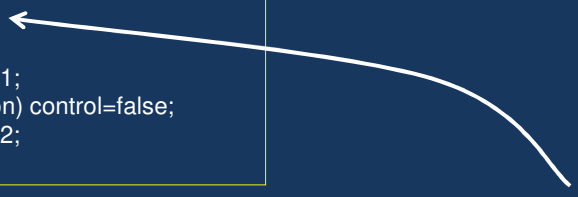
Centinelas y banderas

Bucles controlados por ***banderas (flag)***

Se establece una variable a 1, y cuando la iteración debe acabar, se pone a 0

```
bool control=true;

while (control)
{
      sentencias1;
      if (expresión) control=false;
      sentencias2;
}
```



*Se repite indefinidamente, hasta que **control** se fije a 0*

23

Complemento: números aleatorios en C/C++

```
#include<stdlib.h>
#include<time.h>
```

srand() Establece la semilla del generador de aleatorios

rand() Genera un entero entre [0..RAND_MAX]

```
float A;
int B;
srand(time(NULL));
…
…
A = rand()/(float)RAND_MAX;
B = (rand() % 11)+5;
A = (rand()/(float)RAND_MAX)*10;
A = ((rand()/(float)RAND_MAX)*10 )+5;
```

→ Un float [0..1]
→ Un entero [5..15]
→ Un float [0..10]
→ Un float [5..15]

24

25