CS1112: Programación II

Unidad 1: Introducción al diseño de algoritmos y programación en C++

Sesión de Teoría - 2

Profesor:
José Antonio Fiestas Iquira jfiestas@utec.edu.pe

Material elaborado por:
Maria Hilda Bermejo, José Fiestas, Rubén Rivas

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Índice:

Unidad 1: Introducción al diseño de algoritmos y programación en C++

- Sesión 1:
  - Introducción al curso
  - Comprender el diseño de un algoritmo y la estructura de un programa en C++ (rol de los algoritmos en el proceso de solución de problemas).
  - Conocer y usar tipos de datos primitivos (ej., números, caracteres, booleanos) y operadores
- Sesión 2:
  - Conocer y usar estructuras de control

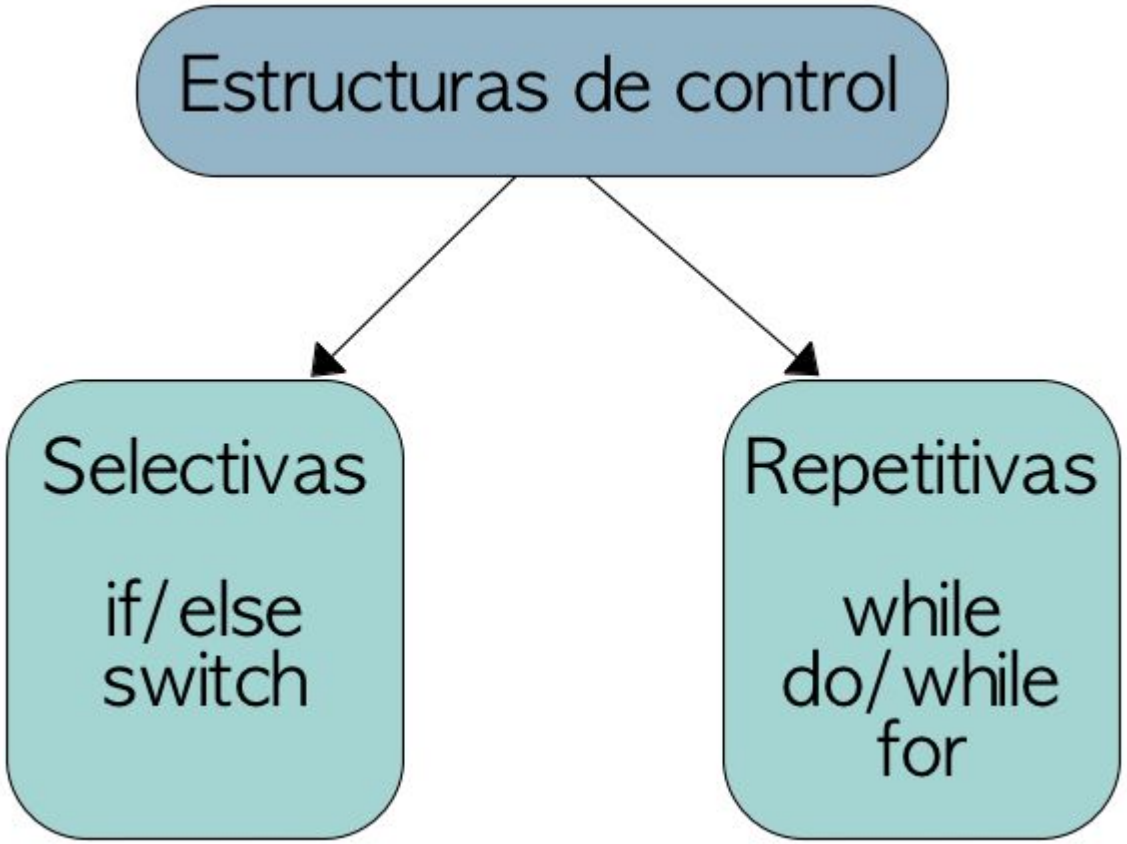
1.4 Estructuras de Control

Logro de la sesión:

Al finalizar la sesión, los alumnos podrán:

- Conocer y usar estructuras de control en C++

Estructuras de Control en C++:



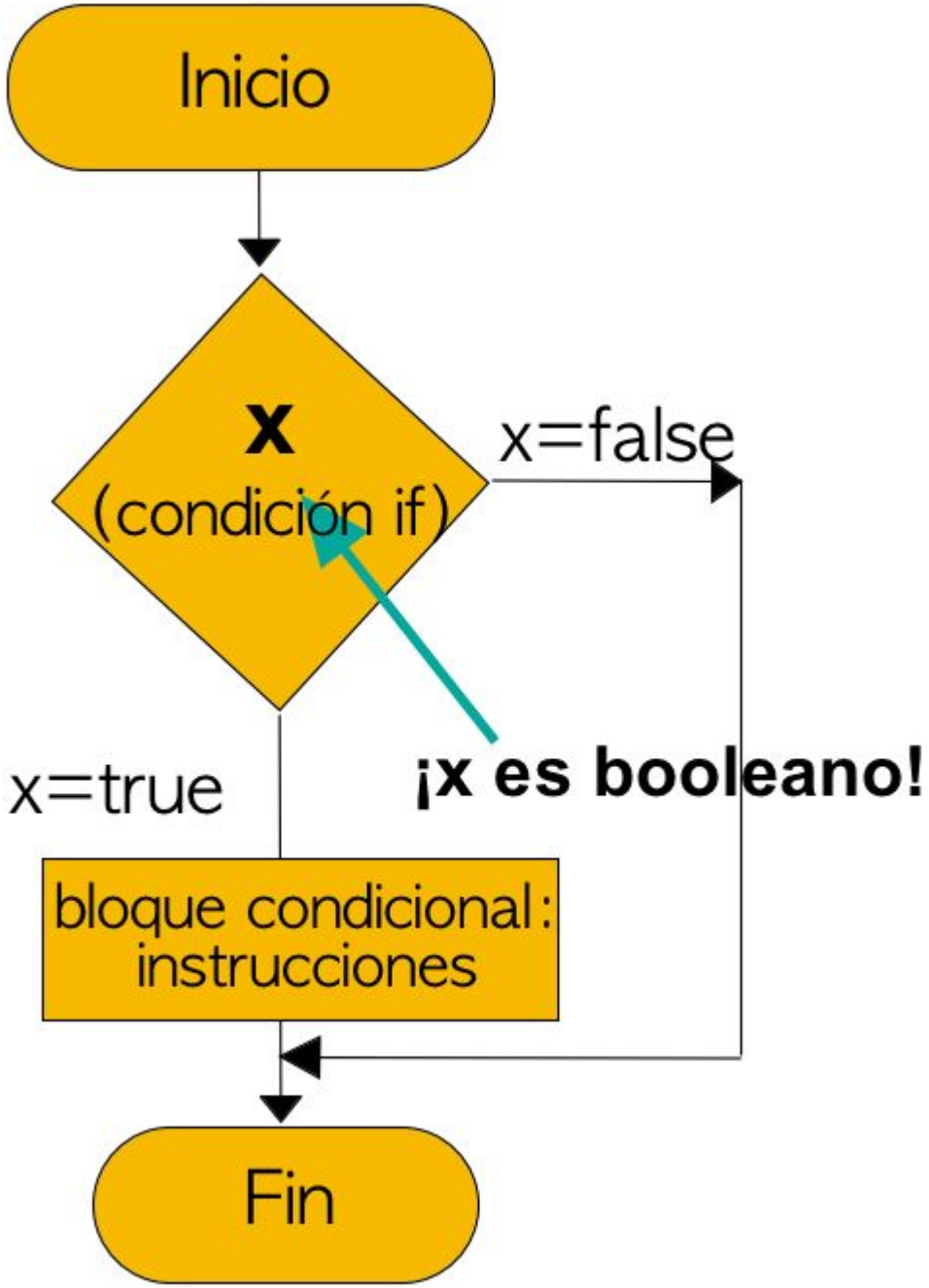
Estructuras de control selectivas:

if
if/else
switch

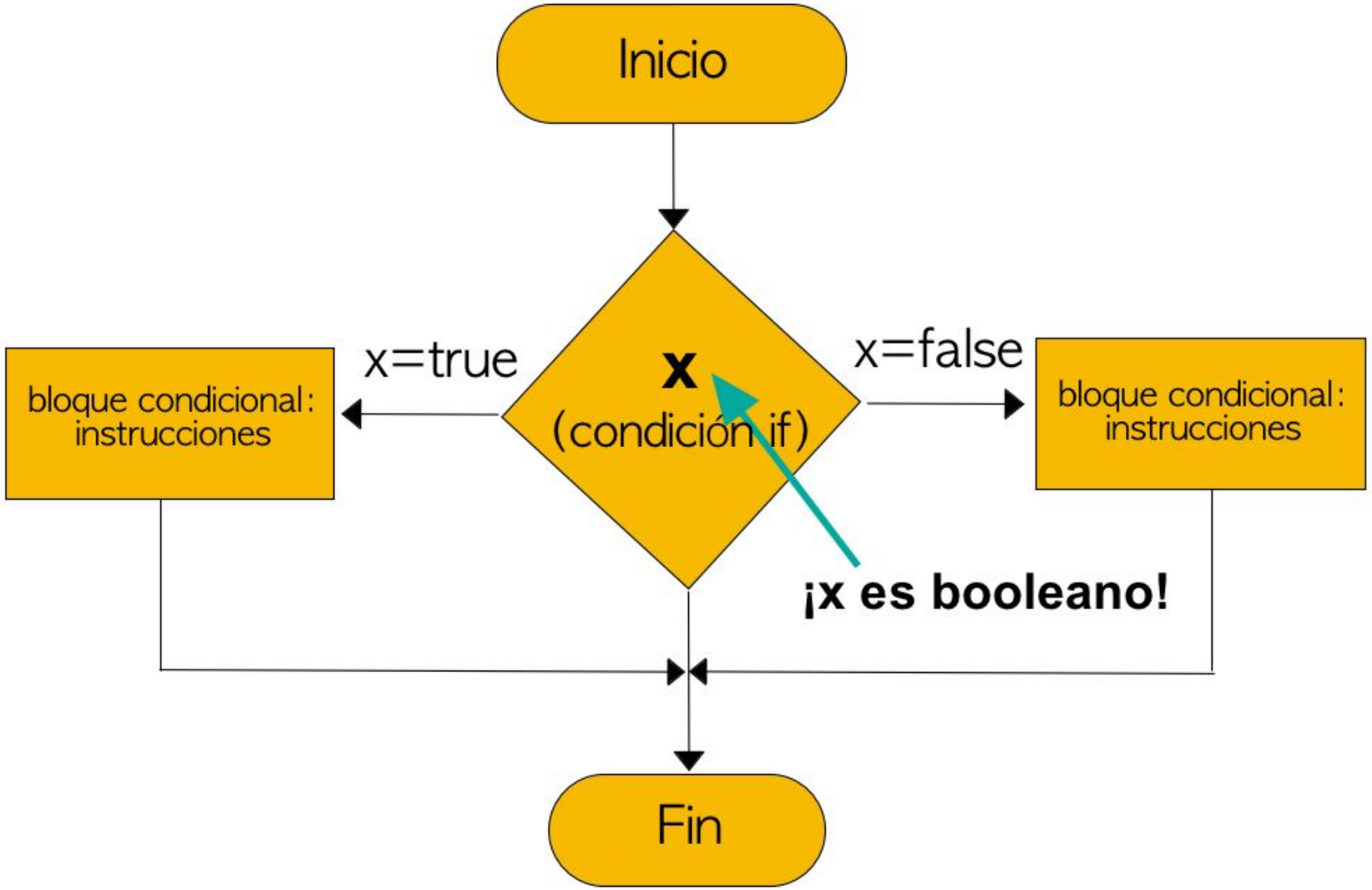
if

```
if ( condición)
    instrucción 1;
```

```
if ( condición)
{
    instrucción 1;
    instrucción 2;
    instrucción 3;
    ....
    instrucción n;
}
```



if / else




```
if (condición)
    instrucción1;
else
    instrucción2;
```

```
if ( condición)
{  bloque condicional:
instrucciones
}
else
{  bloque condicional:
instrucciones
}
```

Ejemplo 01:

Escribir un programa que permita **leer dos números enteros** y los **divida** solo en caso el denominador sea distinto a cero.
Si el denominador es cero, debe señalar que el cociente no está definido

switch

Esta instrucción permite seleccionar un bloque de código a ejecutar según el valor de la expresión.

```
switch(expresión)
{
   case opción1:
            bloque_de_código 1;
   case opción2:
            bloque_de_código 2;
   ...
   default:
             bloque_de_código;
}
```

Los valores de la *expresión* sólo pueden ser de 3 tipos de datos:

- int
- char
- enum

Las opciones tienen que ser datos ordinales, no pueden ser: float, double o string.
Las opciones no pueden ser rangos.

A partir de la opción seleccionada todos los bloques inferiores se ejecutarán al menos que se incluya la instrucción break antes del bloque para detener la ejecución de los bloques posteriores.

```
switch (expresión)
{
    case opcion_a:
            bloque de instrucciones a;
            break; // termina el switch y continua
    case opcion_b:
            bloque de instrucciones b;
            break; // termina el switch y continua
     ...
}
```

Ejemplo 02:

Escribir un programa que permita leer un número entero que podría tener los valores 1, 2, 3 ó 4, e imprima la estación de año que le corresponda según la tabla :

1 Otoño

2 Invierno

3 Primavera

4 Verano

Ahora cambie los enteros por char, con las opciones ‘p’, ‘v’, ‘o’, ‘i’

Solución 1

```
#include <iostream>
using namespace std;
int main()
{ int numero=0;

 cout <<"Numero :";
 cin>>numero;
 switch(numero)
 {
     case 1: cout<<"Otoño\n";
             break;
     case 2: cout<<"Invierno\n";
             break;
     case 3: cout<<"Primavera\n";
             break;
     case 4: cout<<"Verano\n";
             break;
     default:
             cout << "No corresponde a una estación";
 }
 return 0;
}
```

Estructuras de control repetitivas:

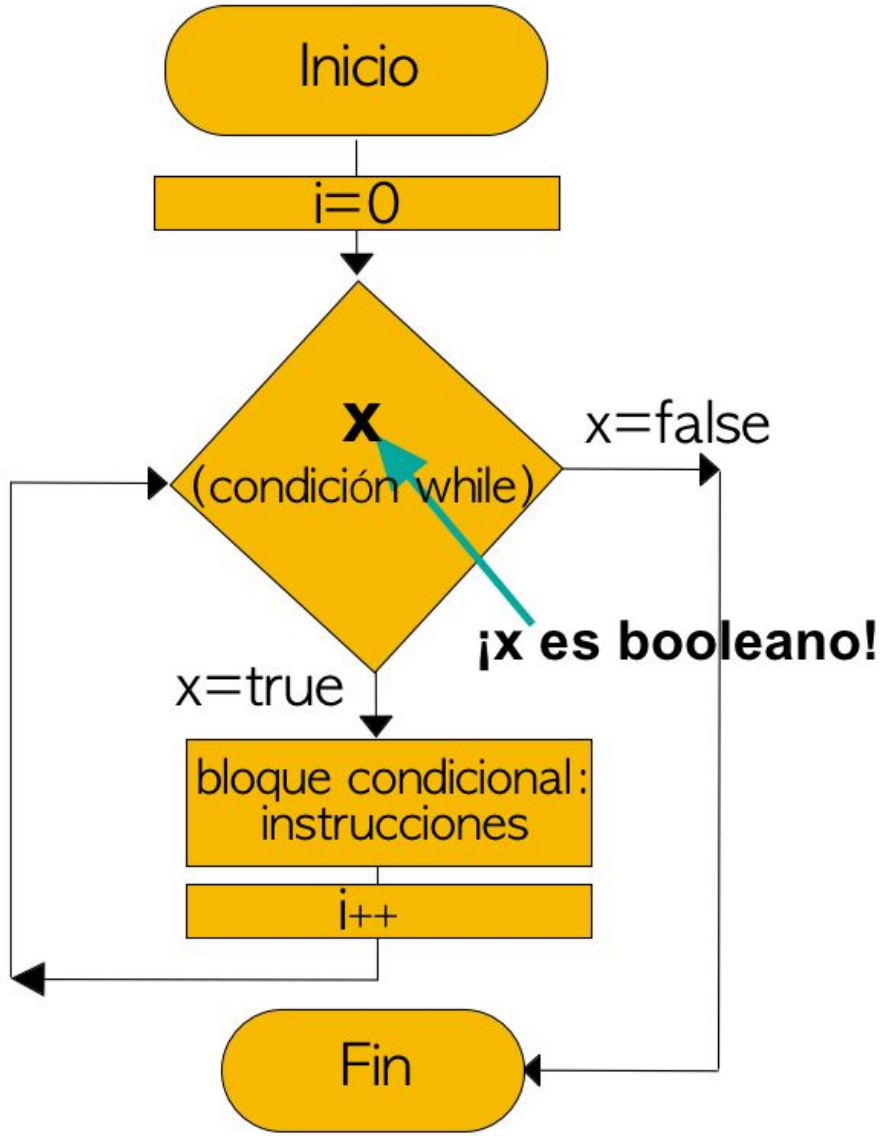
while
do/while
for

while

Permite ejecutar un conjunto de instrucciones repetitivamente, mientras la **expresión lógica** **sea verdadera**.

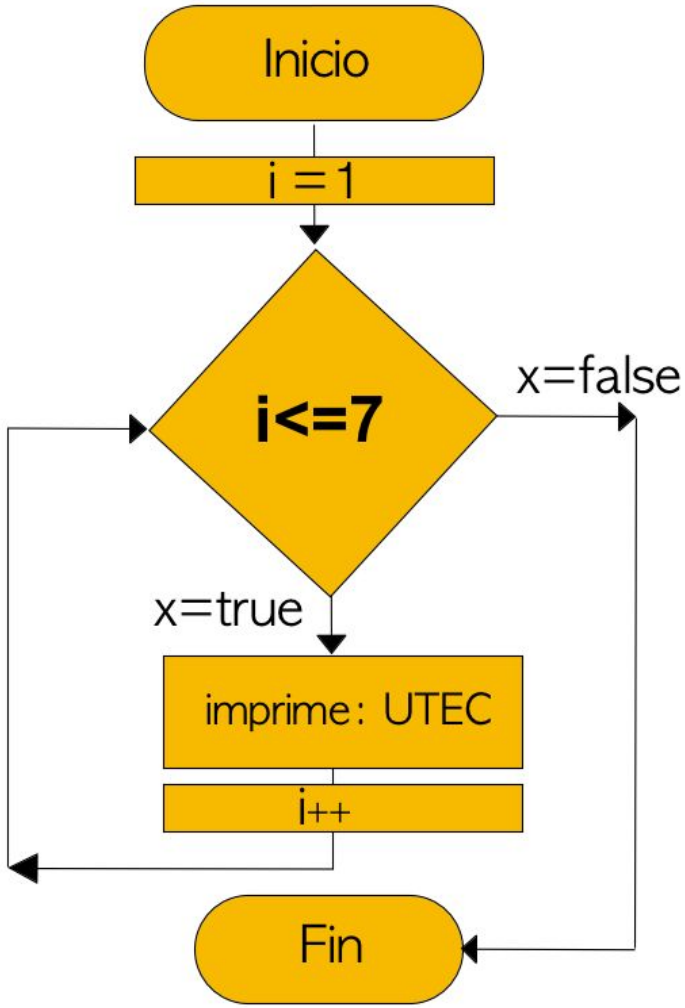
```
while(expresión lógica)
        instrucción;
```

```
while(expresión lógica)
{   instrucción_1;
    instrucción_2;
    instrucción_3;
}
```



Se imprime 7 veces la palabra UTEC, usando el bucle while

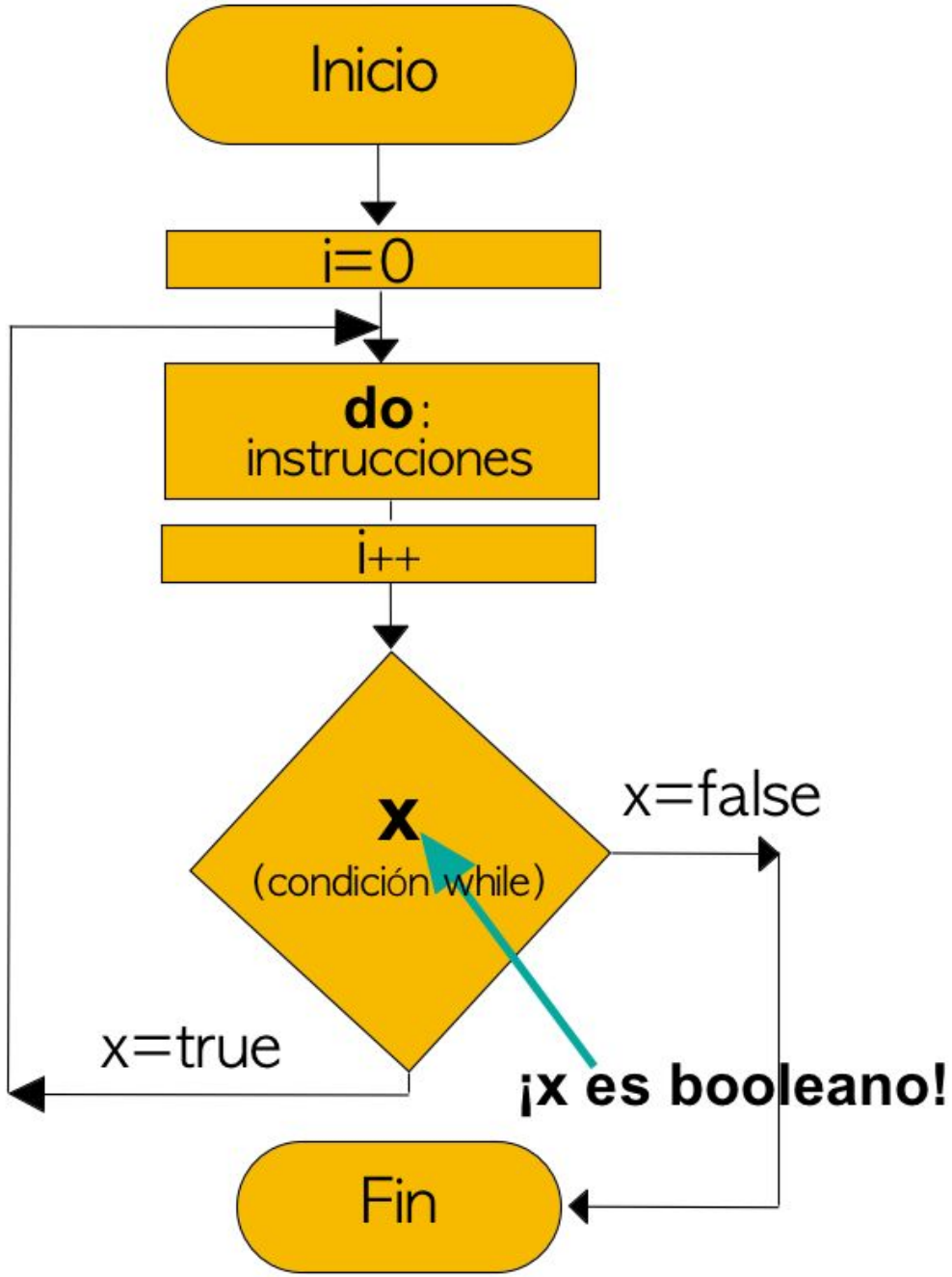
```
#include <iostream>
using namespace std;

int main()
{
        int i;

        i = 1;
        while( i <= 7)
        {
                cout  <<  " UTEC \n";
                i++;
        }

        return(0);
}
```



do while

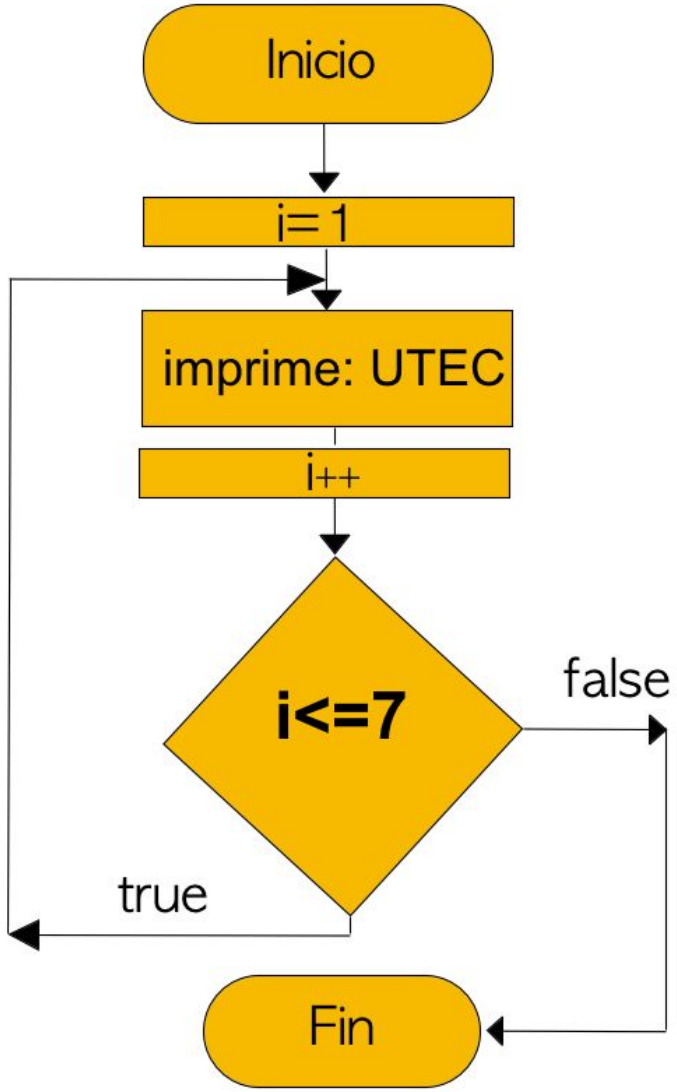
Se ejecuta repetitivamente mientras la **expresión lógica sea verdadera**.
A diferencia de **while,** la **expresión lógica** se verifica al final, por lo tanto el **conjunto de instrucciones** se ejecutará por lo menos **una vez**.

```
do
{    instrucción_1;
     instrucción_2;
     instrucción_3;
  ...
     instruccion_n;
}while(expresion_logica);
```



Se imprime 7 veces la palabra UTEC, utilizando el bucle: do while

```
#include <iostream>
using namespace std;

int main()
{
        int i;

        i = 1;
        do
        {
                cout << "UTEC \n";
                i++;
        }while(i<=7);

        return(0);
}
```

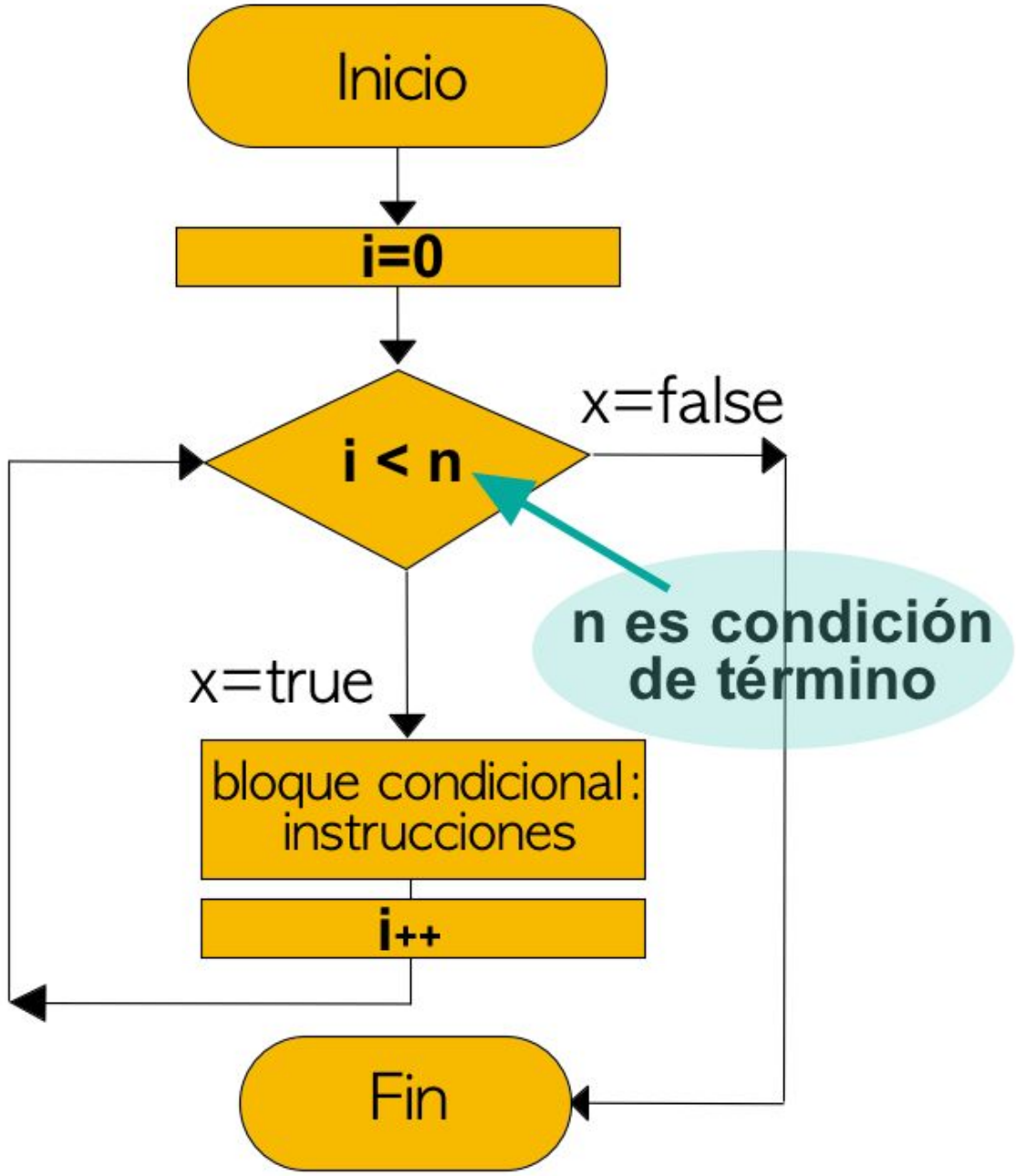


for

Define de **forma compacta** la repetición de un conjunto de instrucciones, incluyendo en la cabecera de la estructura **3 acciones** que usualmente se realizan en un bucle, separadas por punto y coma (;).

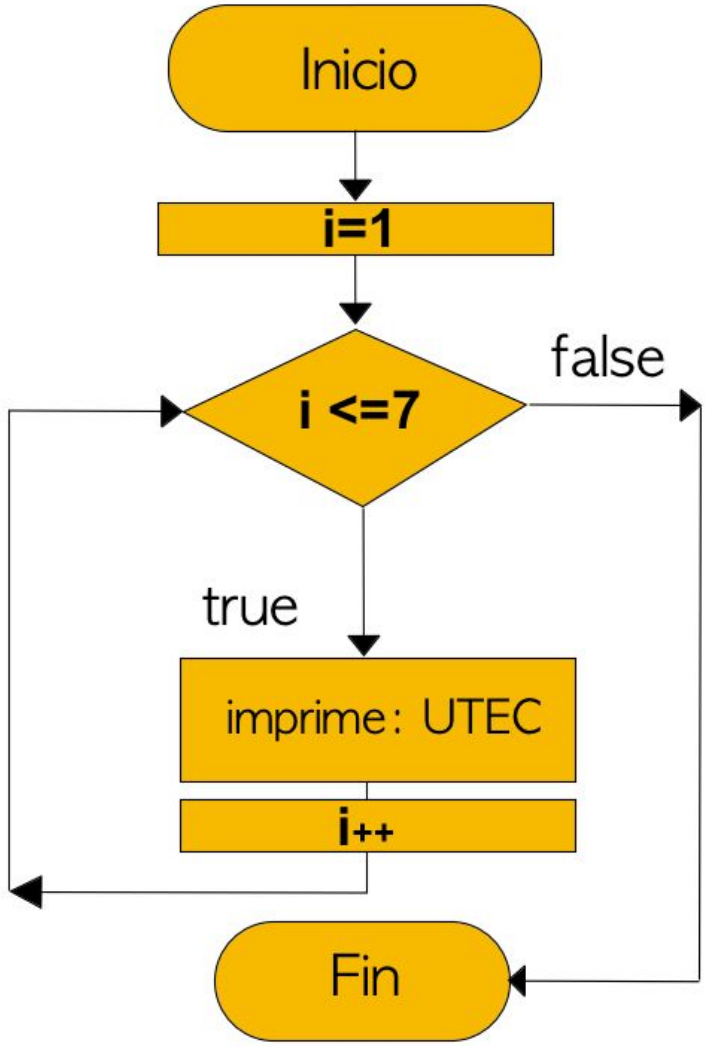
```
for(inicializaciones; condición;  variaciones)
        instruccion_1;
```

```
for(inicializaciones; condición;
variaciones)
{
    instruccion_1;
    instruccion_2;
    ...
    instruccion_n;
}
```



Se imprime 7 veces la palabra UTEC, usando for

```
#include <iostream>
using namespace std;

int main()
{
        for(int i =1; i <=7; i++)
                cout << “UTEC \n”;

        return(0);
}
```



Ejemplo 03:

Comparando los bucles!

Qué ocurre con los programas si en los 3 casos, se cometiera el error de inicializar i=10 ?

```
#include <iostream>
using namespace std;

int main()
{ int i;

 i=10;
 while(i<=7)
 {
   cout << "UTEC\n";
   i++;
 }

 return 0;
}
```

```
#include <iostream>
using namespace std;

int main()
{ int i;

 i=10;
 do{
    cout << "UTEC\n";
    i++;
 }while(i<=7);

 return 0;
}
```

```
#include <iostream>
using namespace std;

int main()
{

 for(int i=10; i<=7; i++)
   cout << "UTEC\n";

 return 0;
}
```

Ejemplo 04:

- Realizando más de una inicialización

```
for(inicializaciones; condición;  variaciones)
        instruccion 1;
```

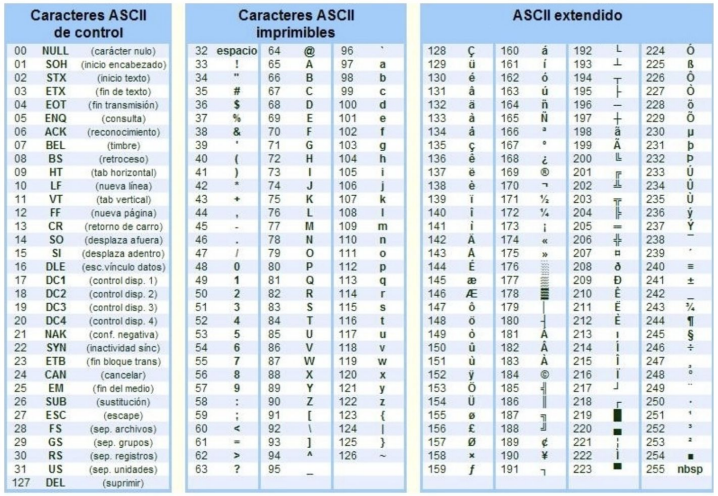
```
//--- Halla la suma de los múltiplos de cinco, desde el 10 al 95
#include <iostream>
using namespace std;

int main()
{ int suma;

for (int contador = 10, suma = 0; contador <= 95;  contador += 5 )
    suma += contador;
 cout << "La sumatoria es " << suma;
 return 0;
}
```

Ejemplo 05:

- Sin usar la parte de la inicialización

```
for(  ; condición;  variaciones)
        instrucción 1;
```

```
//---Imprime de manera descendente el abecedario utilizando letras mayúsculas
#include <iostream>
using namespace std;
int main()
{ char letra='Z';

 for(  ; letra>='A'; letra--)
   cout << letra << " ";
 return 0;
}
```



Caracteres ASCII de control		
00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sinc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

Caracteres ASCII imprimibles					
32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Resumen

1 Selectivas

if
if… else

switch

2 Repetitivas

while

do … while

for

3 Prueba en laboratorio:

break
continue
exit
? :

¡ Prueba estos ejemplos !

Ejemplo:

Escribir un programa, que permita leer como dato un número entero de al menos 7 dígitos y el programa indique si el número es capicúa.

Un número es capicúa si se lee el mismo número, cuando se lee de izquierda a derecha o se lee de derecha a izquierda

Ejemplo 1:
Ingrese número: 1235321
Es número es capicúa

Ejemplo 2:
Ingrese número: 8765342
El número no es capicúa.

main.cpp

```
#include "funciones.h"

int main()
{tipo_Entero numero;

 numero=LeeNumero();
 if(numero == NumeroInvertido(numero))
   cout<<"Es capicua!";
 else
   cout<<"No es capicua!";
 return 0;
}
```

funciones.h

```
#include <iostream>
#include <iostream>
#include "tipos.h"
using namespace std;


tipo_Entero LeeNumero();
tipo_Entero NumeroInvertido(tipo_Entero
num);
```

tipos.h

```
//typedef long int tipo_Entero;
using tipo_Entero = long int;
```

funciones.cpp

```
#include "funciones.h"
tipo_Entero LeeNumero()
{//--------------------
      tipo_Entero n;
      do{
            cout <<"Numero de por lo menos 7 digitos: ";
            cin>>n;
      }while(n<1000000);
      return n;
}

tipo_Entero NumeroInvertido(tipo_Entero num)
{//-------------------------------------
      tipo_Entero numeroAlReves, digito;

      numeroAlReves=0;
      while(num>0)
      {
            digito = num%10;
            numeroAlReves = numeroAlReves*10 + digito;
            num/=10;  //---  num = num/10;
      }
     return numeroAlReves;
}
```

Algoritmo: por Ej si el número fuese: 1234567
numeroAlReves, se va formado asi:

0*10 + 7 = 7
7*10 + 6 = 76
76*10+5 = 765
765*10+4 = 7654
7654*10+3=76543
76543*10+2=765432
765432*10+1 = 7654321

Ejemplo:

En C++ no existe una forma estándar de convertir todo un texto a mayúsculas, minúsculas o colocar la primera letra de cada palabra a mayúsculas (capitalize), C++ brinda solo *toupper* y *tolower* que funciona para un solo caracter, usando for por rangos se puede implementar fácilmente estas funciones.

upper_case
lower_case
capitalize

En C++ no existe una forma estándar de verificar si un texto es entero, C++ brinda sólo la función *isdigit* que funciona para un solo caracter, usando for por rangos se puede implementar una versión simple de esta función.

is_integer

En el ejemplo, también se implementan funciones para remover espacios en blanco antes y despueś de un texto, tales como:

ltrim - Elimina los espacios a la izquierda
rtrim - Elimina los espacios a la derecha
trim - Elimina los espacios a ambos lados

main.cpp

```
#include "funciones.h"
int main()
{
 cout << upper_case("Hello World!\n");
 cout << lower_case("HELLO WORLD!\n");
 cout << capitalize("HELLO WORLD!\n");
 cout << boolalpha << is_integer("+1234") <<
"\n";
 cout << "**" << ltrim("     Hola") << "**\n";
 cout << "**" << rtrim("Hola     ") << "**\n";
 cout << "**" << trim("      Hola     ") <<
"**\n";
 return 0;
}
```

Pantalla de salida:

```
HELLO WORLD!
hello world!
Hello World!
true
**Hola**
**Hola**
**Hola**
```

funciones.h

```
#include <iostream>
#include <string>
#include "tipos.h"
using namespace std;
typedef string Text;
typedef bool Boolean;
Text upper_case(Text text);
Text lower_case(Text text);
Text capitalize(Text text);
Boolean is_integer(Text text);
Text ltrim(Text text);
Text rtrim(Text text);
Text trim(Text text);
```

tipos.h

```
typedef string Text;
typedef bool Boolean;
```

funciones.cpp

```cpp
#include "funciones.h"

Text upper_case(Text text)
{//-----------------------
 Text result;
 for (auto car: text)
   result += toupper(car);
 return result;
}

Text lower_case(Text text)
{//-----------------------
 Text result;
 for (auto car: text)
   result += tolower(car);
 return result;
}
```

```cpp
Text capitalize(Text text)
{//-----------------------
 Text result;
 auto previo = '\0'; // '\0' es el caracter nulo
 for (auto& car: text) {
   if (previo == '\0' || isblank(previo))
     result += toupper(car);
   else
     result += tolower(car);
   previo = car;
 }
 return result;
}

Boolean is_integer(Text text)
{//-------------------------
 Boolean result = false;
 size_t i = 0;
 if (text.size() == 1 && isdigit(text[i]))
   result = true;
 else if (text.size() > 1) {
  result = true;
  if (text[i] == '+' || text[i] == '-')
   ++i;
  for (; i < text.size(); ++i) {
   // Busca el primer carácter que no es dígito
   if (!isdigit(text[i])) {
     result = false;
     break;
   }
  }
 }
 return result;
}
```

```
Text ltrim(Text text)
{//--------------------
 Text result;
 size_t i = 0;
 // Recorre espacios a la izquierda del texto
 // para ubicar primera posición válida
 if (isblank(text[i]))
   while (isblank(text[++i]));

 // Recorre el resto de caracteres y almacena caracteres
 while (text[i] != '\0')
   result += text[i++];

 return result;
}

Text rtrim(Text text)
{//----------------------
 auto result = Text {} ;
 int i = text.size() - 1; // Debe ser entero con signo
 // Recorre espacios a la derecha del texto
 // para ubicar última posición válida
  if (isblank(text[i]))
   while (isblank(text[--i]));

 // Recorre el resto de caracteres y almacena caracteres
 while(i >= 0)
   result = text[i--] + result;
 return result;
}
```

```
Text trim(Text text)
{//-----------------
 return ltrim(rtrim(text));
}
```

Lo aprendido hoy:

- Diferencia entre una estructura selectiva y una estructura repetitiva
- Utilidad de las estructuras de control en un algoritmo

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Bibliografía:

Deitel. P.J. and Deitel. H. M. (2016) C++ How to Program, Prentice Hall.

Stroustrup, Bjarne (2013). The C++ Programming Language, 4th Addison-Wesley.

Eckel, Bruce, 2000. Thinking in C++, Vol 1: Introduction to Standard C++, 2nd Edition, Prentice Hall

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

¡Nos vemos en la siguiente clase!

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA