materia
Informática

unidade didáctica 3

Informática

Efrén Arias Jordán
Departamento de Electrónica e Computación
Escola Politécnica Superior

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
VICERREITORÍA DE ESTUDANTES, CULTURA E FORMACIÓN CONTINUA

titulación
Grao en Enxeñaría Civil

unidade **didáctica 3**

Informática

Efrén Arias Jordán

Departamento de Electrónica e Computación
Escola Politécnica Superior

Deseño
Unidixital
Servizo de Edición Dixital
da Universidade de Santiago de Compostela

Edita
Vicerreitoría de Estudantes,
Cultura e Formación Continua
da Universidade de Santiago de Compostela
Servizo de Publicacións
da Universidade de Santiago de Compostela

Imprime
Unidixital

Dep. Legal: C 333-2013
ISBN 978-84-9887-934-6

MATERIA: Informática
TITULACION: Grao en Enxeñaría Civil
PROGRAMA XERAL DO CURSO
Localización da presente unidade didáctica

Unidade I. Introdución
- Representación da información.

Unidade II. Introdución á programación
- Algoritmos e programas
- Linguaxes de Programación
- Tradutores: compiladores e intérpretes
- Ciclo de vida do software
- Paradigmas de programación

Unidade III. Datos e instrucións
- Estrutura dun programa
- Constantes e variables
- Codificación dos datos: tipos
- Tipos de instrucións
- Subrutinas ou subprogramas
- Librerías
- C++

Unidade IV. Programación estruturada
- Estruturas de control
- Estrutura secuencial
- Estrutura alternativa
- Estrutura repetitiva (bucles)

Unidade V. Programación modular
- Introdución ao deseño estruturado. Concepto de subprograma e función. Programas modulares
- Librerías de funcións en C++
- Funcións na linguaxe C++
- Definición e Chamada
- Prototipo
- Argumentos
- Valor de retorno
- Paso de Parámetros: por valor e por referencia

Unidade VI: Datos estruturados
- Definición Dato estruturado
- Tipo Punteiro
- Vectores estáticos, introdución
- Vectores Unidimensionais
  - Concepto
  - Declaración
  - Uso e percorrido de Vectores Unidimensionais
  - Paso de Vectores Unidimensionais a funcións
- Vectores Bidimensionais ou matrices

- Concepto
- Declaración
- Uso e percorrido de Vectores Bidimensionais
- Paso de Vectores Bidimensionais a funcións

- Vector de carácteres e cadeas de texto
- Vectores multidimensionais
- Estruturas e Unións

Unidade VII. Ficheiros

- Concepto de ficheiros e os seus tipos
- Declaración dunha variable tipo ficheiro
- Operacións con ficheiros:
- Abrir e pechar ficheiros
- Control de erros e fin de ficheiro
- Lectura/escritura de ficheiros de texto
- Lectura/escritura de ficheiros binarios
- O acceso directo á información de arquivos

Unidade VIII. Introdución a MATLAB

INDICE

Presentación 7
Os obxectivos 7
A metodoloxía 8
Os contidos básicos 9
1. Linguaxe C: características 9
2. Operadores 13
3. Programas en C++ 18
4. Estruturas de Control Fundamentais 29
5. Estrutura Secuencial 29
6. Estrutura Alternativa 29
7. Estrutura Iterativa 34
8. Exemplos de Programación 39
Anexo: actividades propostas 56
Avaliación da UD **58**
Bibliografía 58

PRESENTACIÓN

A variedade de plataformas de programación dispoñibles hoxe en día é moi ampla e está en constante evolución. Nesta Unidade Didáctica preséntanse os fundamentos de programación que hai entodas as linguaxes de programación, e en particular en Linguaxe C e C++, para que o alumnado, partindo dun problema sinxelo, sexa capaz de atopar un algoritmo que o resolva. Para acadar iso, mostrarase como desenvolver programas sinxelos e os fundamentos de programación que hai detrás

Móstranse os conceptos básicos, comúns a todas as linguaxes de programación e en particular na linguaxe C e C++, e unha metodoloxía para desenvolver programas de forma sistemática e eficiente.

Así mesmo procúrase introducir os tres tipos básicos de sentenzas de control: secuencia, selección e iteración, explicando en profundidade a súa sintaxe e semántica, de xeito que o alumnado sexa capaz de implementar ditas estruturas de control en linguaxe C ou C++ e os fundamentos da Programación Estruturada.

A formulación desta unidade didáctica é eminentemente práctica, de xeito que o alumnado dispoña dunha ampla variedade de exemplos que lle permita acadar as competencias transversais e específicas da materia na que está inmersa esta UD.

A duración da presente UD e de 15 horas de docencia expositiva e interactiva, deixando a parte de actividades para a resolución das mesmas por parte do alumnado, fora do horario académico

OS OBXECTIVOS

Os obxectivos,cara o alumnado, perseguidos nesta unidade didáctica son:

1) introducirse nos conceptos e técnicas básicos da programación dende un enfoque xeral, que lle permita adaptarse ás distintas linguaxes de programación, en particular o linguaxe C e C++.
2) Adquirir os fundamentos básicos de programación que hai detrás das linguaxes de ultima xeración, e os hábitos e destrezas baseados na programación estruturada.
3) Desenvolver as destrezas necesarias para a resolución de problemas básicos de cálculo no ámbito da enxeñería mediante o uso das ferramentas de programación. Aprender a desenvolver pequenos algoritmos e programas exemplo utilizando as distintas estruturas de control elixindo o tipo das sentenzas adecuadas para cada caso, en función das características particulares do programa a desenvolver.
4) Ser capaz, ante unha nova especificación dun problema, de adoptar unha unha solución cos recursos dispoñibles e o de abstraer o coñecemento aprendido para poder utilizalo nun novo ámbito

A METODOLOXÍA

A metodoloxía de ensinanza que se persigue dentro da materia onde se encadra a presente unidades didáctica e a seguinte:

Clases Teóricase Prácticas

As clases consistirán na explicación dos apartados do programa coa axuda dunha presentación electrónica. Tamén se realizarán exercicios na pizarra, facendo que o alumnadodesenvolva un programa ditando ao profesor as sentenzas que eles propoñen para a resolución do exercicio. As diapositivas da presentación serán postas a disposición do alumnado.

As clases fundamentalmente terán lugar nunha aula de informática, na que se proporcionará un ordenador para cada alumno. A metodoloxía de aprendizaxe de prácticas consiste fundamentalmente na resolución das actividades propostas e outros exercicios de programación, individualmente ou por grupos, coa axuda do profesor.

Titorías

As sesións de titorías servirán para resolver as dúbidas do alumnado en canto ós contidos da materia, resolución de problemas de teoría e exercicios de prácticas propostos no anexo de actividades. Estas titorías serán tanto presenciais como virtuais a través da plataforma virtual, e son fundamentais para acadar unha aprendizaxe efectiva da materia.

Curso Virtual

Esta materia dispoñerá dun curso virtual desenvolto sobre unha plataforma de e-learning. Nela facilitaráselle ó alumnado todo o material necesario en formato dixital, ademais de distintas ferramentas de comunicación para o apoio ás titorías, incluíndo correo electrónico e foros.

Linguaxe C: características

- Linguaxe de programación de propósito xeral, moi adecuado para programación de sistemas (UNIX foi escrito en C).
- Linguaxe relativamente pequena: só ofrece sentenzas de control sinxelas e funcións.
- A E/S non forma parte da linguaxe, senón que se proporciona a través dunha biblioteca de funcións.
- Permite a agrupación de instrucións.
- Programación estruturada.
- Permite a separación dun programa en módulos que admiten compilación independente.
- Deseño modular.
- Programas portables.
- Non é unha linguaxe fortemente tipada.
- É bastante permisivo coa conversión de datos.
- Sen unha programación metódica pode ser propenso a erros difíciles de atopar.
- A versatilidade de C permite crear programas difíciles de ler.

```
#define _ -F<00||--F-OO--; int F=00,OO=00;
main(){F_OO();printf("%1.3f\n",4.*-
F/OO/OO);}F_OO() {…..}
```

/* Exemplo 1. Programa DUCIA en linguaxe C */

```
#include <stdio.h>
main ()
{
int ducia;
      ducia = 12;
      printf ("Unha ducia son %d unidades\n",
      ducia);
}
```

En C++

```
#include <iostream>
using namespace std;
int main ()
{
      int ducia=12;
      cout<<"Unha ducia son:"<<ducia<<"
      unidades"<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
```

```
}
```

Basicamente o C está composto polos seguintes elementos:

- Comentarios
- Identificadores
- Palabras reservadas
- Variables e tipos de datos
- Constantes
- Operadores

Palabras Reservadas

auto	**Do**	**for**	**return**	**typedef**
break	**double**	**goto**	**short**	**union**
case	**else**	**if**	**sizeof**	**unsigned**
char	**enum**	**int**	**static**	**void**
const	**extern**	**long**	**struct**	**volatile**
continue	**float**	**register**	**switch**	**while**
default				

É preciso insistir en que C fai distinción entre maiúsculas e minúsculas. Por tanto, a palabra reservada **`for`** non pode escribirse como FOR, pois o compilador non a recoñece como unha instrución, senón que a interpreta como un nome de variable.

Variable

Identificador utilizado para representar un certo tipo de información.Cada variable é dun tipo de dato determinado. Unha variable pode almacenar diferentes valores en distintas partes do programa.

Tipos de datos Básicos

TIPO	PALABRA RESERVADA	TAMAÑO EN PALABRAS
sen valor	**void**	0
Carácter	**char**	1
Enteiro	**int**	2
coma flotante (simple precisión	**float**	4
coma flotante (dobre precisión	**double**	8

Declaración de Variables

Unha declaración asocia un tipo de datos determinado a unha ou máis variables. O formato dunha declaración é:

```
tipo_de_dato var1, var2, ..., varN;
```

Exemplos:

```
int a, b, c;
float numero_1, numero_2; char letra;
unsigned long enteiro;
```

Deben declararse todas as variables antes do seu uso. Deben asignarse aos variables nomes significativos.

```
int temperatura;
int k;
```

A función printf()

Permite imprimir información pola saída estándar (pantalla)

```
printf (<cadea de control>, <resto argumentos>);
printf(formato, argumentos);
```

Exemplos:

```
printf("Ola mundo\n");
printf("O número 28 é %d\n", 28);
printf("Imprimir %c %d %f\n", 'a', 28, 3.0e+8);
printf (“páxina %d \n var = %f”, 5, estatura * 2);
```

Cadea de control
Contido
Secuencias de escape: \n \t \\ \”
Comandos de formato (mesmos que scanf)
Resto dos carácteres

Do mesmo xeito que en scanf teñen que coincidir o número de ‘%’ e o número de variables

Formatos da función printf()

TIPO DE ARGUMENTO	CARÁCTER DE FORMATO	FORMATO DE SAÍDA
Numérico	%d	signed decimal int
	%i	signed decimal int
	%o	unsigned octal int
	%u	unsigned decimal int

	%x	unsigned hexadecimal int (con a, ..., f)
	%X	unsigned hexadecimal int (con A, ..., F)
	%f	[-]dddd.dddd
	%e	[-]d.dddd ou ben e[+/-]ddd
	%g	o máis curto de %e e %f
	%E	[-]d.dddd ou ben E[+/-]ddd
	%G	o máis curto de %E e %f
Carácter	%c	carácter simple
	%s	cadea de carácteres
	%%	o carácter %
Punteiros	%n	
	%p	

Secuencias de escape da función printf()

CARÁCTER	SIGNIFICADO
\a	Alarma (Beep)
\b	Retroceso (BS)
\t	Tabulador Horizontal (HT)
\n	Nova Liña (LF)
\v	Tabulador Vertical (VT)
\f	Nova Páxina (FF)
\r	Retorno
\"	Comiñas dobres
\'	Comiña simple
\\	Barra investida

A función printf()

Especificadores de ancho de campo: A sentenza

```
printf("Numero enteiro = %5d \n", 28);
```

produce a saída:

```
Numero enteiro =_ _ _28
```

A sentenza

```
printf("Numero real = %5.4f \n", 28.2);
```

produce a saída:

Numero real = 28.2000

Función scanf()

Permite ler datos do usuario. A función devolve o número de datos que se leron ben.

```
scanf (<cadea de control>, <resto argumentos>);
scanf(formato, argumentos);
```

Os especificadores de formato igual que printf().

Exemplos:

```
scanf("%f", &numero);
scanf("%c\n", &letra);
scanf("%f %d %c", &real, &enteiro, &letra);
scanf("%ld", &enteiro_longo);
scanf("%s", cadea);
char inicial; int idade; float estatura;
scanf(“%c %d %f”, &inicial, &idade, &estatura);
scanf(“%c %d”, &inicial, &idade, &estatura);
```

Entrada/ Saída de carácteres

```
int getchar(void); // Devolve carácter ou EOF
int putchar(int); // Devolve parámetro ou EOF
```

Exemplo:

```
#include <stdio.h>
int main(void)
{
      int c;      // Coidado
      c = getchar();
      while (c != EOF)
      {
            putchar(c);
            c = getchar();
      }
      return 1;
}
```

Mellor escrito con outro estilo:

```
while ( (c = getchar()) != EOF) putchar(c);
```

Operadores
Operador de asignación

Forma xeral:

```
identificador = expresión ;
```

Exemplos:

```
a = 3;
area = lado * lado;
```

O operador de asignación = e o de igualdade ==

Asignacións múltiples:

```
ide_1 = ide_2 = ... = expresión
```

As asignacións efectúanse de dereita a esquerda.

En i = j = 5

1. A j asígnaselle 5
2. A i asígnaselle o valor de j

Conversión de tipos

(tipo de dato) expresión

```
int x; float e;
x = 5;
e = x / 2;
```

o valor asignado á variable **e** será 2, pois / realiza unha división enteira.

```
int x; float e;
x = 5;
e = (float) x / 2;
```

Agora, a variable e almacena o valor 2.5.

```
int x; float e;
x = 5;
e = (float) (x / 2);
```

non se asigna a e o valor 2.5, senón 2, xa que as parénteses que envolven á expresión x / 2 fan que primeiro se efectúe a división enteira e logo a conversión a flotante

Conversións entre Tipos simples
Implícitas

```
int i = 8; double d;
d = i;
d = 5.9;
i = d;
```

Explícitas

```
(tipo) expr
(int) 5.4
i = (int) d;
```

Exemplo

```
double x;
int a = 5, b = 2;
x = a / b;
```

Operadores aritméticos

División enteira (/): división dunha cantidade enteira por outra, desprézase a parte decimal do cociente.

O operador % require que os dous operandos sexan enteiros.A maioría das versións de C asignan ao resto o mesmo signo do primeiro operando. Os valores negativos co signo –

	OPERADOR	DESCRICIÓN
UNARIOS	-	Cambio de signo
	--	Decremento
	++	Incremento
BINARIOS	-	Resta
	+	Suma
	*	Produto
	/	División
	%	Resto división enteira

Operadores incremento, decremento

Se o operador segue ao operando o valor do operando modificarase despois da súa utilización. Se o operador precede ao operando o valor do operando modificarase antes da súa utilización.

A expresión	é equivalente a
i++;	i =i+1
++i;	i =i+1
i--;	i =i-1
--i;	i =i-1

se a = 1	Imprime:
printf("a = %d \n", a);	**a = 1**
printf("a = %d \n", ++a);	**a = 2**
printf("a = %d \n", a++);	**a = 2**
printf("a = %d \n", a);	**a = 3**

Operadores relacionais

Utilízanse para formar expresións lóxicas. O resultado é un valor enteiro que pode ser:

- **certo**, represéntase cun 1
- **falso, represéntase cun 0**

OPERADOR	DESCRICIÓN
>	Maior que
>=	Maior ou igual que
<	Menor que
<=	Menor ou igual que
==	Igual que
!=	Diferente que

Exemplo 1: Se a = 1 e b = 2

Expresión	Valor	Interpretación
a < b	1	certo
a >b	0	falso
(a + b) != 3	0	falso
a == b	0	falso
a == 1	1	certo

Operadores lóxicos

Actúan sobre operandos que son á súa vez expresións lóxicas que se interpretan como:

- **certo**, calquera valor distinto de 0
- **falso**, o valor 0

	OPERADOR	DESCRICIÓN
UNARIOS	**!**	**not**
BINARIOS	**&&**	**and**
	||	**or**

a	B	!a	a && b	a || b
F	F	V	F	F
F	V	V	F	V
V	F	F	F	V
V	V	F	V	V

Exemplo

&& e avalíanse de esquerda a dereita.
! avalíase de dereita a esquerda.
Exemplos: se a = 7 e b = 3

Expresión	Valor	Interpretación
(a + b) < 10	**0**	**falso**
!((a + b) < 10)	**1**	**certo**
(a ! = 2) || ((a +b) <= 10)	**1**	**certo**
(a 4) >&& (b < 5)	**1**	**certo**

Operadores de asignación compostos

O operador de asignación pódese combinar con outros operadores como ***, /, %, +, -, <<, >>, &, |, ^*** para operacións acumulativas

m *= 5;	**m = m * 5;**
m += b;	**m = m + b;**
m += e - 3;	**m = m + e - 3;**
m -= (e = 5);	**m = m - (e = 5);**

Precedencia das Operacións aritmético/lóxicas

Avaliación a igual nivel	Nivel de prioridade	Operadores
→	1º	**( )[]->**
←	2º	**++ -- (cast) & sizeof**
→	3º	***/ %**
→	4º	**+ -**
→	5º	**<<>>**
→	6º	**<<=>>=**
→	7º	**== !=**
→	8º	**&**
→	9º	**^**
→	10º	**|**
→	11º	**&&**
→	12º	**||**
←	13º	**?:**
←	14º	**operadores deasignación**
→	15º	**,**

Programas en C++

En C++calquera programa é unha función

```
int main(void)
```

Devolvemos un valor (que podemos usar para indicar se houbo erros)

```
return 0;
```

Para mostrar mensaxes en C++ úsase ***cout***

```
cout << “Ola, mundo” << endl;
```

O texto que se vai mostrar vai entre comiñas. Os símbolos **<<** son o operador de inserción. Para terminar a liña usamos ***endl*ou *n***
As sentenzas en C++terminan en ;

O obxecto cout

O obxecto ***cout***representa a saída estándar (normalmente, a pantalla ou consola). Este obxecto declárase no arquivo de cabeceira ***iostream***

```
#include <iostream>
```

Vive no espazo de nomes ***std***polo quedeberiamos escribir ***cout::std.*** Para evitalo podemos utilizar a directiva ***using***

```
using namespace std;
```

Exemplo

```
#include <iostream>
using namespace std;
int main(void)
{
      cout << "Ola, mundo" << endl;
      return 0;
}
```

Entrada en C++.

Para obter datos en C++ úsase ***cin***

```
int a;
cin >> a;
```

Lese da entrada estándar (normalmente, o teclado). Os símbolos **>>** son o operador de extracción. Espérase un dato do mesmo tipo que a variable que se usa.
Pódense ler varios datos seguidos

```
int a,b;
cin >> a >> b;
```

Un programa completo en C++

```
#include <iostream>
using namespace std;
int main(void)
{
```

```
        int nota1, nota2;
        float media;
        cout << "Introduce a nota do primeiro exame"
        << endl;
        cin >> nota1;
        cout << "Introduce a nota do segundo exame"
        << endl;
        cin >> nota2;
        media = (nota1+nota2)/2;
        cout << "A media é" << media << endl;
        return 0;
}
```

Variables en C++

Unha variable é o elemento dun programa que almacena un valor que pode cambiar durante a execución. Cada variable é dun tipo que indica que valores pode almacenar
Os tipos simples en C++ son

- **`char`**
- **`int`**
- **`float`**
- **`double`**
- **`bool`**

Algúns deles pódense modificar con ***`long`, `short`, `signed`, `unsigned`***

Tipos, valores e rangos en C++

Tipo	Exemplos	Tamaño/Rango (*)
`char`	‘a’, ’A’, ’5’, ’$’, ’ ‘	1 byte
`int`	4, -815, 1623, -42, 0	4 bytes [-2147483648, 2147483647]
`float`	3.1416, -4.715, 108, 2E+8, 3.14E-7	4 bytes +/-3.4E+/-38 (7 díxitos)
`double`	igual que float	8 bytes +/-1.7E+/-308 (15 díxitos)
`bool`	true, false	[true, false]

* Dependen da arquitectura da máquina. Móstranse valores típicos para un sistema de 32 bits

Nomes de variable en C++

Cada variable ten un nome, que pode estar formado por

- letras
- díxitos
- guión baixo ou subliñado “_”

O nome dunha variable non pode empezar por un díxito

Exemplos

- media
- nota1
- x
- a miña_variable

As letras maiúsculas e minúsculas son diferentes, é dicir, ***media***e ***Media***son variables distintas

Declaración e asignación de variables en C++

Antes de usar unha variable, hai que declarala indicando o seu tipo e o seu nome

```
int nota1;
float media;
bool aprobado;
```

O valor dunha variable establécese mediante unha sentenza de asignación

```
nota1 = 5;
aprobado = true;
```

Declaración e asignación pódense combinar na mesma sentenza

```
int nota1 = 5;
bool aprobado = true;
```

Un programa en C++ con variables

```
#include <iostream>
using namespace std;
int main(void)
{
      int nota = 5;
      cout << ”A túa nota é " << nota << endl;
      nota = 10;
      cout << ”Agora a túa nota é " << nota <<
      endl;
      return 0;
}
```

Operadores en C++

Os valores das variables pódense combinar mediante operadores

```
int valor1 = 5;
int valor2 = 3;
int suma = valor1 + valor2;
```

Operadores aritméticos binarios: +, -, *, /, %
Operadores aritméticos unarios: ++, --, -
Operadores lóxicos: &&, ||, !
Operadores relacionais: <, <=, >, >=, ==, !=

Exemplos con operadores en C++

```
int a,b,c;
a=3;
b=2;
c=a+b*3;          // c vale 9
c=(a+b)*3;        // c vale 15
c=b++;            // c vale 2, b vale 3
c=++a;            // c vale 4, a vale 4

int a,b,c;
float x;
bool ou;
a=3;
b=2;
c=(a+b)/2;        // c vale 2
x=(a+b)/2.0;      // x vale 2.5
ou=x>c;           // ou vale true
```

Precedencia e asociatividade de operadores

Operador	Asociatividade
()	De esquerda a dereita
++ -- - !	De dereita a esquerda
*** / %**	De esquerda a dereita
+ -	De esquerda a dereita
<<=>=>	De esquerda a dereita
&&	De esquerda a dereita

||	De esquerda a dereita
== !=	De esquerda a dereita
=	De dereita a esquerda

Exemplos en C e en C++

```
#include <stdio.h>
#define MENSAXE “alumnos”
/*prototipo funciones*/
int imprimir_saudo (char *destino);
int main (void)
{
      int erro=0;
      erro=imprimir_saudo (MENSAXE); return(0);
}
int imprimir_saudo (char *destino)
{
      printf (“ola %s \n", destino); return(0);
}
```

En C++...

```
#include <cstdlib>
#include <iostream>
#define MENSAXE "alumnos"
using namespace std;
int imprimir_saudo (char *destino)
{
      cout<<"Ola "<<destino<<endl;
      return (0);
      system ("Pause");
}
int main()
{
      int erro=0;
      erro=imprimir_saudo (MENSAXE);
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Programa que le os lados dun rectángulo e calcula a súa área.

```
#include <stdio.h>
int main(void)
{
      float base, altura;
      printf(“Teclee datos: “);
```

```
        scanf (“%f %f”, &base, &altura);
        printf(“A superficie é %f \n”, base *
        altura);
        return 1;
}
```

En C++

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
        float base,altura;
        cout<<"Introducir a base:";
        cin>>base;
        cout<<"Introducir a altura:";
        cin>>altura;
        cout<<"A superficie é:"<<base*altura<<endl;
        system("PAUSE");
        return EXIT_SUCCESS;
}
```

Exemplo: Facer un programa que lea un número enteiro e ou eleva ao cadrado:

```
#include <stdio.h>
void main()
{
        int numero;
        int cadrado; // mellor que fose long int
        printf("Introduza un numero:");
        scanf("%d", &numero);
        cadrado = numero * numero;
        printf("O cadrado de %d
        é%d\n",numero,cadrado);
}
```

En C++

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
        int numero;
        long int cadrado;
```

```
        cout<<"Introducir un numero:";
        cin>>numero;
        cadrado=numero*numero;
        cout<<"O cadrado de:"<<numero<<"
        é:"<<cadrado<<endl;
        system("PAUSE");
        return EXIT_SUCCESS;
}
```

Programa que le o raio dun círculo e calcula a súa área.

```
#include <stdio.h>
#define PI 3.141593
void main()
{
        float radio;
        float area;
        printf("Introduza o radio: ");
        scanf("%f", &radio);
        area= PI*radio*radio;
        printf("O area do círculo é %5.4f \n",
        area);
}
```

En C++

```
#include <cstdlib>
#include <iostream>
#define PI 3.141593
using namespace std;
int main()
{
        float radio,area;
        cout<<"Introducir o radio:";
        cin>>radio;
        area=PI*radio*radio;
        cout<<"A áreado círculo é:"<<area<<endl;
        system("PAUSE");
        return EXIT_SUCCESS;
}
```

Sexan dous cadrados de lados L1 e L2 inscritos un noutro. Calcula a área da zona comprendida entre ambos, utilizando para iso unha función (que se chamará Cadrado) que devolve a área dun cadrado cuxo lado se pasa como argumento

```
#include <stdio.h>
int cadrado (int);
int main (void)
```

```
{
      int lado2, lado1;
      printf ("dáme lado do cadrado 1:");
      scanf("%d",&lado1);
      printf ("dáme lado do cadrado 2:");
      scanf("%d",&lado2);
      printf ("O resultado da diferenza é %d",
      cadrado(lado1)-cadrado(lado2));
      return(0);
}
int cadrado (int lado)
{
      return(lado*lado);
}
```

En C++

```
#include <cstdlib>
#include <iostream>
using namespace std;
int cadrado (int);
int main()
{
      int lado2,lado1;
      cout<<"Dáme o lado 1:";
      cin>>lado1;
      cout<<"Dáme o lado 2:";
      cin>>lado2;
      cout<<"O resultado da diferenza de cadrados
      é:"<<cadrado(lado1)-cadrado(lado2)<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
}
int cadrado (int lado)
{
      return (lado*lado);
}
```

Programa que converte graos Fahrenheit a graos centígrados.
C = (5/9) * (F - 32)

```
#include <stdio.h>
void main()
{
      float centigrados;
      float fahrenheit;
```

```
        printf("Introduza unha temperatura en graos
        fahrenheit: ");
        scanf("%f", &fahrenheit);
        centigrados = 5.0/9 * (fahrenheit - 32);
        printf("%f graos fahrenheit = %f graos
        centigrados \n", fahrenheit,centigrados);
}
```

En C++

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
        float farenheit, centigrados;
        cout<<"introducir unha temperatura en graos
        Farenheit:";
        cin>>farenheit;
        centigrados=5.0/9*(farenheit-32);
        cout<<farenheit<<" graos Farenheit son
        "<<centigrados<<" graos centigrados"<<endl;
        system("PAUSE");
        return EXIT_SUCCESS;
}
```

Este programa converte a graos Fahrenheit unhatemperatura que vén en graos Centígrados. Cámbiaopara que faga a conversión inversa.

```
#include <iostream>
using namespace std;
int main()
{
        // A relación entre as temperaturas
        centígradas e fahrenheit é C/100=(F-32)/180
        const float F1=32;      /* Constante */
        const float F2=180;     /* Constante */
        const float C1=100;     /* Constante */
        float C, F;       /* Variable*/
        cout << "Introduce unha temperatura en graos
        centígrados: ";
        cin >> C;
        F=C/C1*F2+F1;
        cout << "A temperatura en graos fahrenheit é
        ";
        cout << F << endl;
        system("pause");
        return 0;
}
```

Este programa pide por teclado as notas de catro probas dunha materia e escribe '1' se aprobaches e '0' se suspendiches, de dúas formas distintas. Son equivalentes as dúas expresións? Próbao coas notas 4,5,4,6. Engade unha terceira forma de calcular a media para que, neste último caso, obteñas un 4.75

```
#include <iostream>
using namespace std;
int main()
{
      int nota1, nota2, nota3, nota4;
      float media1, media2, cuadrimestre1,
      cuadrimestre2;
      bool aprobado1, aprobado2;
      const float limite = 4.5;
      cout << "Introduce a nota do primeiro
      parcial : "; cin >> nota1;
      cout << "Introduce a nota do segundo
      parcial: "; cin >> nota2;
      cout << "Introduce a nota do terceiro
      parcial : "; cin >> nota3;
      cout << "Introduce a nota do cuarto parcial
      : "; cin >> nota4;
      media1 = (nota1+nota2+nota3+nota4)/4;
      cuadrimestre1 = (nota1+nota2)/2;
      cuadrimestre2 = (nota3+nota4)/2;
      media2 = (cuadrimestre1+cuadrimestre2)/2;
      cout << "Calculando a túa media da primeira
      forma, tes un " << media1 << endl;
      cout << "Calculando a túa media da segunda
      forma, tes un " << media2 << endl;
      aprobado1 = (media1 >= limite);
      aprobado2 = (media2 >= limite);
      cout << "Agora móstrase un 1 se aprobaches,
      ou un 0 se non: " << endl;
      cout << "Da primeira forma: " << aprobado1
      << endl;
      cout << "Da segunda forma: " << aprobado2 <<
      endl;
      system("pause");
      return 0;
}
```

Dados os números enteiros a, b, c, d, deséxase:
En primeiro lugar sumar a+b, dividir o resultado obtido entre c e á cantidade obtida restarlle d;
Algún dos valores almacenados na variable "res" é correcto?. Proporciona a solución correcta se ningún dos tres propostas o é.

```
#include <iostream>
using namespace std;
int main()
{
      int a, b, c, d;
      cout << "Introduce 4 números enteiros " <<
      endl;
      //Aquí debes executar a orde para almacenar
      o catro valores nas variables a, b, c, d.
      cin >> a>> b>>c >> d;
      float res=0;
      res= a+b / c-d;
      cout << "O resultado desta operación é" <<
      res << endl;
      res= a+ (b/c) -d;
      cout << "O resultado desta operación é" <<
      res << endl;
      res= (a+b)/(c-d);
      cout << "O resultado desta operación é" <<
      res << endl;
      system("pause");
      return 0;
}
```

Este programa está totalmente finalizado e está pensado para exercitarse no bo uso da precedencia e asociatividade dos operadores. Nel aparecen unha serie de expresións que debes analizar para determinar o valor de cada unha delas. Seguidamente executa o programa para comprobar que os teus resultados son correctos.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
      int a=1,b=2,c=3;
      cout << a + b * c << endl;
      cout << (a + b) *c << endl;
      a = 1; b = 2;
      a = b = a += b * 3;
      cout << a ' << ' << b << endl;
      a = 1; b = 2;
      a = b = (a += b) * 3;
```

```
        cout << a ' << ' << b << endl;
        a = 1; b = 2; c = 3;
        cout << a + b % c + c / b * c - b * b + a *
        b * c % a endl;. <<
        system("PAUSE");
        return 0;
}
```

Estruturas de control fundamentais

Existen 3 estruturas de control fundamentais:

- Secuencial(**BLOCK**)
- Alternativa (**IF-THEN-ELSE**)
- Repetitiva (Existen 2 tipos: **WHILE** e **DO-WHILE**)

Estrutura Secuencial

Consiste en executar unha sentenza a continuación doutra.Pódense agrupar entre chaves varias sentenzas para formar unha única sentenza, denominada “sentenza composta”, da seguinte maneira:

```
{
sent1;
sent2;
sent3;
...
sentN;
}
```

Estrutura alternativa

Permite elixir entre dúas alternativas, segundo sexa *verdadeira* ou *falsa*, a condición que se avalía.Existen dous subtipos

Alternativa simple(**if**)

Alternativa dobre(**if-else**)

Se a “condición” é verdadeira execútase a “sentenza”. Se non, non se executa ningunha acción.

Implementación alternativa simple

```
if (cond)
sent;
```

No caso de que haxa máis dunha “sentenza”, deberanse de incluír entre chaves, para formar unha única sentenza composta.

Exemplo

Supondo que a variable nota de tipo float contén a cualificación dun alumno, o seguinte fragmento de programa determina se superou a proba.

```
float nota;
// O programa obtén a nota por algún medio
if (nota >= 5.0)
      cout << " Proba superada ";
```

Se a "**condición**" é verdadeira execútase a "**sentenza1**". Se non, execútase a "**sentenza2**".

Implementación alternativa dobre

```
if (cond)
      sent1;
else
      sent2;
```

Tanto "**sent1**" como "**sent2**" deben de ser unha única sentenza. Se houbese varias sentenzas en calquera das dúas ramas, deberanse de incluír entre chaves, para formar unha única sentenza composta.

Exemplo

O seguinte fragmento de programa determina se a variable **num** de tipo **int** contén un número par ou impar.

```
int num; // O programa inicializa num por algún
medio
if (num%2 == 0)
      cout << " É par ";
else
      cout << " É impar ";
```

Exemplo `if - else`

Programa que le un número e di se é par ou impar.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      int numero;
      cout<<"Introducir un numero:";
      cin>>numero;
      if (numero%2==0)
            cout<<"O numero é par"<<endl;
      else
            cout<<"O numero é impar"<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
```

```
}
```

Tipo da condición

```
if (a > b)
if (27)
if (a+b)
if (func())
```

Erros non detectados

```
int a = 6;
if (a = 5)
      printf (“É cinco”);
else
      printf (“É seis”);
```

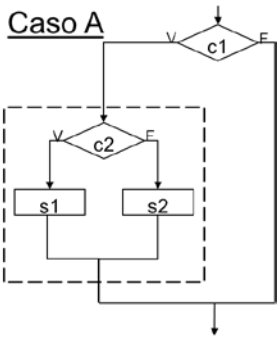
Aniñamento de estruturas alternativas

```
if (c1)
      if (c2)
            s1;
      else
            s2;
else
      if (c3)
            s3;
      else
            s4;
```

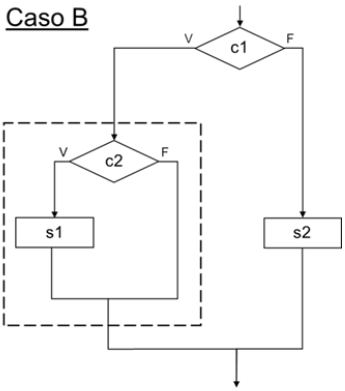
Ambigüidade da cláusula else

Prodúcese cando, en estruturas aniñadas, existen máis cláusulas **`if`**que **`else`**.Iso pode ser debido, por exemplo, ao aniñamento de estruturas **`if`**en estruturas**`if-else`**, ou viceversa.

A continuación móstranse dous casos que, aínda que representan 2 situacións diferentes, con todo conducen ao mesmo código.




```
if (c1)
     if (c2)
          s1;
     else
          s2;
```




```
if (c1)
     if (c2)
          s1;
     else
          s2;
```

Para desfacer esta ambigüidade tómase o convenio de emparellar cada **else**co if máis próximo.No exemplo anterior o código resultante corresponde ao caso A.E entón… como se codificaría o caso-B?

```
if (c1)
{
      if (c2)
            s1;
}
else
      s2;
```

Estrutura alternativa múltiple (switch)

Variable debe ser tipo enteiro ou letra

```
switch (variable) {
      case cte1: sentenza;
            break;
      case cte2: sentenza;
            break;
      default: sentenza;
}
```

Exemploswitch

Solución 1:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      char letra;
      cout<<"Introduza unha letra:";
      cin>>letra;
      switch (letra)
      {
            case 'a':
            case 'A':
                  cout<<"Vocal a ou A"<<endl;
                  break;
            case 'e':
            case 'E':
                  cout<<"Vocal e ou E"<<endl;
                  break;
            case 'i':
            case 'I':
                  cout<<"Vocal i ou I"<<endl;
                  break;
            case 'o':
            case 'O':
                  cout<<"Vocal o ou O"<<endl;
                  break;
            case 'u':
            case 'U':
                  cout<<"Vocal u ou U"<<endl;
                  break;
            default:
                  cout<<"Consonante"<<endl;
      }
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Solución 2:

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      char letra;
      cout<<"Introduza unha letra:";
```

```
        cin>>letra;
        switch (letra)
        {
              case 'a':
              case 'A':
              case 'e':
              case 'E':
              case 'i':
              case 'I':
              case 'o':
              case 'O':
              case 'u':
              case 'U':
                    cout<<"Vocal"<<endl;
                    break;
              default:cout<<"Consonante"<<endl;
        }
        system("PAUSE");
        return EXIT_SUCCESS;
  }
```

Estrutura Iterativa

Instrución `while`

```
        while (expresión)
              sentenza;
```

A "**condición**" avalíase ao principio, é dicir primeiro avalíase expresión, polo que o bucle execútase 0 ou máis veces.

sentenza executarase mentres o valor de expresión sexa verdadeiro (distinto de 0). Mentres a "**condición**" sexa verdadeira execútase a "**sentenza**". O normal é que **sentenza**inclúa algún elemento que altere o valor de expresión, proporcionando a condición de saída do bucle. Se a **sentenza** é composta encérrase entre { }

Exemplo

O seguinte bucle acumula a suma de todos os números introducidos por teclado. Finaliza ao meter o cero.

```
        int num, suma;
        suma=0;
        cout <<"Introduza un número.(Cero para acabar): ";
        cin >> num;
        while (num != 0)
        {     suma=suma+num;
              cout << "Seguinte número: ";
              cin >> num;
        }
        cout << "A suma vale " << suma << endl;
```

Programa que le un número N e calcula 1 + 2 + 3 + … + N

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      int N,suma,i;
      cout<<"introducir un numero:";
      cin>>N;
      suma=0;
      i=1;
      while (i<=N)
      {
            suma=suma+i;
            i++;
      }
      cout<<"O valor da suma desde 1 ate "<<N<<"
      é:"<<suma<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Instrución do-while

```
do
{
      sentenza;
}while (expresión);
```

A "**sentenza**" execútase mentres a "**condición**" sexa verdadeira, é dicir executarase mentres o valor de expresiónsexa verdadeiro (distinto de 0).

sentenza sempre se executa polo menos unha vez (diferente a while). O bucle execútase 1 ou máis veces. A "**condición**" avalíase ao final do bucle.O normal é que **sentenza** inclúa algún elemento que altere o valor de expresión, proporcionando a condición de saída do bucle.

Para a maioría das aplicacións é mellor e máis natural comprobar a condición antes de executar o bucle (**while**).

Este bucle sempre leva chaves. En consecuencia pode haber varias sentenzas dentro do bucle.

Exemplo

O seguinte bucle obriga ao usuario a introducir un número comprendido entre 1 e 5, incluídos ambos. (É un uso típico deste tipo de bucle).

```
int num;
do{
      cout << "Introduza un nº comprendido entre 1
      e 5: ";
      cin >> num;
} while (num < 1 || num > 5);
```

Programa que le de forma repetida un número e indica se é par ou impar. O programa repítese mentres o número sexa distinto de cero.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      int N;
      do
      {
            cout<<"Introducir un numero:";
            cin>>N;
            if (N%2==0)
                  cout<<"O numero "<<N<<" é
                  par"<<endl;
            else
                  cout<<"O numero "<<N<<" é
                  impar"<<endl;
      }while (N=!0);
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Bucle controlado por contador(`for`)

```
for (inicialización; condición; progresión)
sentenza;
```

`inicialización`: inicialízase algún parámetro que controla a repetición do bucle. Execútase unha soa vez antes de entrar no bucle.

`condición`: é unha condición que debe ser certa para que se execute **`sentenza`**. Mentres a "**`condición`**" sexa verdadeira execútase a "**`sentenza`**" do bucle. O bucle repítese mentres **condición** non sexa cero (falso). Se se omite **`condición`** asumirase o valor permanente de 1 (certo) e o bucle executarase de forma indefinida.

`Progresión`: utilízase para modificar o valor do parámetro e execútase ao final de cada iteración.

Se **`sentenza`**é composta se encerra entre { }.

`Inicialización`e **`progresión`** pódense omitir.

Quizá a forma máis fácil de entender o bucle **`for`**sexa utilizando a construción **`while`** equivalente:

```
for (inicialización; condición; progresión)
      sentenza;
```

É equivalente a:

```
inicializacion;
while (condicion)
{
      sentenza;
      progresion;
}
```

Programa que imprime os 100 primeiros números

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      int numero;
      for (numero=0; numero <100; numero++)
            cout<<"numero "<<numero<<n;
      return EXIT_SUCCES;
}
```

Exemplo

O seguinte bucle escribe os números comprendidos entre 1 e 10, ambos os incluídos, un en cada liña e en orde ascendente.

```
int i;
for (i = 1; i <= 10; i++)
      cout << i << endl;
```

Exemplo

O seguinte bucle escribe os números comprendidos entre 10 e 1, ambos incluídos, un en cada liña e en orde descendente.

```
int i;
for (i = 10; i >= 1; i--)
      cout << i << endl;
```

Sentenza `break`

Utilízase para terminar a execución de bucles ou saír dunha sentenza switch. É necesaria na sentenza switch para transferir o control fóra da mesma.

En caso de bucles aniñados, o control transfírese fóra da sentenza máis interna na que se atope, pero non fóra das externas. Pode ser útil cando se detectan erros ou condicións anormais.
Non é aconsellable o uso desta sentenza en bucles pois é contrario á programación estruturada.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      int opcion;
      cout<< "1 - Mensaxe pantalla"<<endl;
      cout<< "2 – saír "<<endl;
      cout<<" calquera carácter, non fai
      nada"<<endl;
      cout<<"Elixa a opción: "<<endl;
      cin>>opcion;
      while (opcion)
      {
            switch(opcion)
            {
            case 1:
                  cout<<"Ola,
                  seleccioneopción"<<opcion<<endl;
                  break;
            case 2:
                  cout<<"Adios "<<endl;
                  return;
                  break;
            default:
                  cout<<"Selección invalida. Ténteo
                  de novo"<<endl;
            } /** salto do break
            cout<<"Elixa a opción, 0 para
            saír"<<endl;
            cin>>opcion;
      }
}
```

Sentenza continue

Esta sentenza utilízase nos bucles **for**, **while** e **do_while.** Cando se executa, forza un novo ciclo do bucle, saltándose calquera sentenza posterior.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
```

```
{
      int n;
      int positivos = 0;
      do
      {
            cout<<"Teclea un numero (-99
            finaliza): "<<endl;
            cin>>n;
            if (n<=0)continue;
            positivos++;
      }while (n!=-99);
      cout<<"Has tecleado "<<positivos<<"números
      positivos"<<endl;
      system("PAUSE");
      return EXIT_SUCCESS
}
```

A sentenza **positivos++** só se executa cando n é un número positivo. Se n é negativo ou vale 0, execútase **continue** que forza unha nova avaliación da condición de saída do bucle.

Exemplos de Programación

Escriba un programa que calcule x^n, sendo x e n dous números que se introducen por teclado

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      int n,x,i;
      double potencia=1;
      cout<<"Dáme x e n: "<<endl;
      cin>>x>>n;
      for (i=0; i<n ; i++)
            potencia=potencia * x;
      cout<<"O resultado é:” <<potencia<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Escriba un programa que calcule e imprima a suma dos pares e dos impares comprendidos entre dous valores que se piden por teclado (x e n)

```
#include <cstdlib>
#include <iostream>
```

```
using namespace std;
int main()
{
      int x,n,i;
      int par=0, impar=0;
      do
      {
            cout<<"Solicito dous números x<n
            "<<endl;
            cout<<"Dáme x: "<<endl;
            cin>>x;
            cout<<"Dáme n: "<<endl;
            cin>>n;
      }while (x>n);
      for(i=x; i<=n; i++ )
            if ((i%2)==0)
                  par+=i;
            else
                  impar+=i;
      cout<<"O resultado suma par: "<<par<<" e
      impar: "<<impar<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Fragmento de programa que calcula o factorial dun número lido por teclado.

n!= n·(n-1)·(n-2)·…·2·1 (se n>0)
n!= 1(se n=0)

```
int n, i, fact;   //debería de ser un float?
do
{
      cout<<"Introduza un nº positivo ou cero : ";
      cin >> n;
} while (n < 0); // Impedimos que “n” sexa
negativo
fact=1;
for (i = 1; i <= n; i++)      //e porque non desde
i=2?
      fact=fact*i;
cout << n << "!= " << fact << endl;
```

Fragmento de programa que calcula o semifactorial dun número lido por teclado.

n!!= n·(n-2)·(n-4)·… ·1 (se n>0)
n!!= 1(se n=0)

```
int n, i, semifact;
```

```
do
{
      cout<<"Introduza un nº positivo ou cero : ";
      cin >> n;
} while (n < 0); // Impedimos que “n” sexa
negativo
semifact=1;
for (i = n; i >= 1; i=i-2)
      semifact=semifact*i;
cout << n << "!!= " << semifact << endl;
```

Os bucles pódense aniñar pero é importante estruturalos de forma correcta.

Exemplo: Calcular 1 + 2 + …N mentres N sexa distinto de 0.

```
#include <cstdlib>
#include <iostream>
using namespace std;
int main()
{
      int N, suma, j;
      do{
            cout<<"Introduza N: "<<endl; cin>>N;
            suma = 0;
            for (j=0;j<=N;j++)
                  suma=suma+j;
            cout<<"1+2+...+N="<<suma<<endl;
      }while(N!=0);
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Fragmento de programa que imprime un rectángulo de 4 filas x 5 columnas de asteriscos.

```
int i, j; // i= filas, j=columnas
for (i = 1; i <=4; i++)
{
      for (j = 1; j <=5; j++)
            cout << "*";
            cout << endl; //Salta de liña tras
            imprimir cada fila
}
```

Programa que lea tres números enteiros (tmp, b e c). Se tmp é menor que b, entón incrementar c nunha unidade. Finalmente, visualizar tódolos valores.

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

int main ()
{
      // Declaración de variables
      int tmp, b, c;
      // Lectura de datos
      cout<< "Introduza valor para tmp --> ";
      cin >> tmp;
      cout<< "Introduza valor para b --> ";
      cin >> b;
      cout<< "Introduza valor para c --> ";
      cin >> c;
      // Estrutura condicional
      if (tmp < b)
            c=c+1;// Equivalente a c++
      // Visualización de resultados
      cout << "Los valores finais son tmp--> "<<
      tmp << ", b --> " << b << " y c --> " << c
      << endl;
      system(“pause”);
      return 0;
}
```

Programa que lea tres números enteiros (tmp, b y c). Se tmp é menor que b, entón incrementar c nunha unidade, senón se tmp é maior que b entón reducir c nunha unidade. Finalmente, visualizar tódolos valores.

```
#include <iostream>
using namespace std;
int main ()
{
      // Declaración de variables
      int tmp, b, c;
      // Lectura de datos
      cout<< "Introduza valor para tmp --> ";
      cin >> tmp;
      cout<< "\nIntroduza valor para b --> ";
      cin >> b;
      cout<< "\nIntroduza valor para c --> ";
      cin >> c;
      // Estrutura condicional
      if (tmp < b)
            c=c+1; // Equivalente a c++;
      else
            if (tmp > b)
                  c=c-1; // Equivalente a c--;
      // Visualización de resultados
```

```
        cout << "\nOs valores finais son tmp--> "<<
        tmp<< ", b --> " << b  << " y c --> " << c
        << endl;
        system(“pause”);
        return 0;
}
```

Programa que lea dous números enteiros (a e b). A continuación lerase un número enteiro dende o teclado almacenándoo en tmp. Se tmp é menor que a increméntase b en 1, senón redúcese b en 1. Repetirase o proceso mentres o valor lido tmp sexa menor que a. Finalmente, visualizar os valores tmp, a e b.

```
#include <iostream>
using namespace std;
int main ()
      {// Declaración de variables
      int tmp, a, b;
      // Lectura de datos a y b
      cout<< "Introduza valor para a --> ";
      cin >> a;
      cout<< "\nIntroduza valor para b --> ";
      cin >> b;
      /*IMPORTANTE: INCIALICIZACIÓN DE tmp para
      que entre no bucle while*/
      cout<< "\nIntroduza un valor para tmp --> ";
      cin >> tmp;
      while (tmp < a)
      {
            // Estrutura condicional
            if (tmp < a){
                  b=b+1; // Equivalente a b++;
            }
            else b=b-1; // Equivalente a b--;
                  cout<< "Introduza un novo valor
                  para tmp --> ";
            cin >> tmp;
      }
      // Visualización de resultados
      cout << "\nOs valores finais son tmp--> "<<
      tmp << ", a --> " << a << " y b --> " << b
      << endl;
      system(“pause”);
      return 0;
}
```

Programa que lea tres números enteiros: a, b e c. A continuación calcular a^c. Finalmente indicar se a^c é maior, menor ou igual que b.

```
#include <iostream>
using namespace std;
int main ()
{
      // Declaración de variables
      int a, b, c;
      double potencia;
      // Lectura de datos a, c y b
      cout<< "Introduza valor para a --> ";
      cin >> a;
      cout<< "\nIntroduza valor para c --> ";
      cin >> c;
      cout<< "\nIntroduza valor para b --> ";
      cin >> b;
      // IMPORTANTE A INICIALIZACION DE potencia
      potencia = 1;
      for (int i =1; i<=c; i++)
            potencia *= a; // Equivalente a
            potencia = potencia * a;
      // Estrutura condicional
      if (potencia > b) // a^c é maior que b
            cout << potencia <<" é maior que b "
            << b << endl;
      else
            if (potencia < b) // a^c é menor que b
                  cout << potencia <<" é menor que
                  b "<< b << endl;
            else
                  cout << potencia <<" é igual a b
                  "<< b << endl;
      system("pause");
      return 0;
}
```

Programa que permita determinar a parte enteira do $\log_a b$.

Partir do programa anterior, e aproveitar o posible do resto. Ler só **a** e **b**, inicializar **c** (será a parte enteira do**$\log_a b$**).Utilizar unha estrutura iterativa **while** para buscar a solución:

Calculamos a^c utilizando un bucle de iteración. Se **a^c** é menor que **b**, ¡quedamos curtos!, incrementamos **c** e volvemos iterar, senón, se **a^c** é igual a **b**, ¡ben!, **c** é a solución buscada, senón (**a^c**é maior que **b**), ¡pasámonos de enteiro!, **c-1** é a solución buscada.

```
#include <iostream>
using namespace std;
int main ()
```

```
{
	// Declaración de variables
	int a, b, c;
	double potencia;
	// Lectura de datos a y b
	cout<< "Introduza valor para b --> ";
	cin >> b;
	cout<< "\nIntroduza valor para a (base) -->
	";
	cin >> a;
	// OLLO coa inicialización de:
		// c (que será o valor do logaritmo
		que estamos procurando)
		// potencia (que será o valor de a^c)
	c=0;
	potencia=1;
	while (potencia < b)
	{
		// IMPORTANTE A INICIALIZACION DE
		potencia EN CADA ITERACION
		potencia = 1;
		// calculo a^c
		for (int i =1; i<=c; i++)
			potencia *= a;
		// Estrutura condicional
		if (potencia > b)
		{
		// c xa é maior que o log, así que
		quedámonos co valor anterior
			if (c>0)
				c=c-1; //Equivalente a c--;
		}
		else
			if (potencia < b)
			//c aínda é menor que o log, así
			que incrementámolo
				c=c+1; //Equivalente a c++;
	}
	// Visualización de resultados
	cout << "\nOs valores finais son a --> " <<
	a << ", b --> " << b << ", ultima potencia
	calculada --> " << potencia<< " e o log_a b é "
	<< c << endl;
	system(“pause”);
	return 0;
}
```

Calcular as raíces dunha ecuación de 2º grao cunha incógnita, $Ax^2+Bx+C=0$, dados seus coeficientes.

```
#include<iostream>
#include<math.h>
using namespace std;
int main () {
      double a,b,c,x1,x2;
      double raiz;
      cout << "Introduza os coeficientes da
      ecuación Ax2 + Bx + C = 0" << endl;
      cout << "A? ";
      cin >> a;
      cout << "B? ";
      cin >> b;
      cout << "C? ";
      cin >> c;
      raiz=b*b-4*a*c;
      x1=(-b+sqrt(raiz))/(2*a);
      x2=(-b-sqrt(raiz))/(2*a);
      cout << "Solucións:X1=" << x1 << "yX2=" <<
      x2 << endl;
      system(“pause”);
      return 0;
}
```

Consideracións

Á hora de realizar cálculos é aconsellable (en ausencia de outras restricións como memoria,…) declarar as variables do tipo que acade maior precisión: **`double`.**

Cómpre o uso de paréntesesnas fórmulas que asegure a orde correcta das operacións (“/(2*a)”).

O algoritmo debe de resolver tódolos casos, e non só no caso xeral: este non dá unha resposta correcta para ecuacións con raíces complexas (ex.: x2+x+1=0).

Debe impedirse que se realicen operacións erróneas: divisións por 0, raíz cadrada dun número negativo,…

Resolver o problema anterior para tódolos casos.

```
#include<iostream>
#include<math.h>
using namespace std;
int main () {
      double a,b,c,x1,x2;
      double raiz;
      cout << "Introduza os coeficientes da
      ecuacion Ax2 + Bx + C = 0" << endl;
```

```
        cout << "A? ";cin >> a;
        cout << "B? ";cin >>b ;
        cout << "C? ";cin >> c;
        if (a!=0)
        {
              raiz=b*b-4*a*c;
              if (raiz<0)
                    cout << "Ecuación sen solucións
                    reais!" << endl;
              else
              {
                    x1=(-b+sqrt(raiz))/(2*a);
                    x2=(-b-sqrt(raiz))/(2*a);
                    cout << “X1=" << x1 << "yX2=" <<
                    x2 << endl;
              }
        }
        else {//a==0
              if (b!=0)
              {
                    x1=-c/b;
                    cout << "Ecuación de 1er grado Bx
                    + C = 0" << endl;
                    cout << "Solución:X1=" << x1 <<
                    endl;
              }
              else //a==b==0
                    cout << "Non é una ecuación!!" <<
                    endl;
        }
        system(“pause”);
        return 0;
  }
```

Dados dous números enteiros, rexistrados nas variables a e b, determinar a súa relación de orde.

Utilizando soamente sentenzas**if**(técnica non recomendable)

```
#include <iostream>
using namespace std;
int main () {
      //Decláranse as variables que utilizaremos
      int a, b;
      /* Pídense os enteiros a y b */
      cout << "Introducir dous enteiros.“ << endl;
      cout << "a: "; cin >> a;
      cout << "b: "; cin >> b;
```

```
        /* Sábese a relación de orde, usando só
        sentenzas if */
        if (a > b)
              cout << "a > b, xa que " << a << " > "
              << b << endl;
        if (a < b)
              cout << "a < b, xa que " << a << " < "
              << b << endl;
        if (a == b)
              cout << "a = b, xa que " << a << " = "
              << b << endl;
        system(“pause”);
        return 0;
}
```

Utilizando só sentenzas**if-else**(técnica recomendable)

```
#include <iostream>
using namespace std;
int main()
{
        //Decláranse as variables que utilizaremos
        int a, b;
        /* Pídense os enteiros a y b */
        cout << "Introducir dous enteiros.“ << endl;
        cout << "a: "; cin >> a;
        cout << "b: "; cin >> b;
        /* Sábese a relación de orde, usando só dúas
        sentenzas if-else */
        if (a > b)
              cout << "a > b, xa que " << a << " > "
              << b << endl;
        else
              if (a < b)
                    cout << "a < b, xa que " << a <<
                    " < " << b << endl;
              else
                    cout << "a = b, xa que " << a <<
                    " = " << b << endl;
        system(“pause”);
        return 0;
}
```

¿Por que é mellor o segundo método que o primeiro?

Co primeiro método sempre se realizan tres comparacións. En cambio, co segundo só se realiza unha, ou ben dúas, no caso máis desfavorable.

¿Por que non se utilizaron chaves de apertura e cerre nos **if**, e nos **if-else?**

Porque soamente se executa unha sentenza en cada caso.
Vexamos a solución do problema anterior, con dúas instrucións a realizar en cada **if**.

```
#include <iostream>
using namespace std;
int main ()
{
    /*Decláranse as variables que utilizaremos*/
    int a, b;
    /* Pídense os enteiros a y b*/
    cout << "Introducir dous enteiros.“<< endl;
    cout << "a: ";cin >> a;
    cout << "b: ";cin >> b;
    /* Sábese a relación de orde, usando soamente
        dúas sentenzas if-else */
    if (a > b)
    {
        cout << "a > b, xa que ";
        cout << a << " > " << b << endl;
    }
    else
    if (a < b)
    {
        cout << "a < b, xa que";
        cout << a << " < " << b << endl;
    }
    else {
        cout << "a = b, xa que";
        cout << a << " = " << b << endl;
    }
    system(“pause”);
    return 0;
}
```

Se nas variables a e b temos rexistrados dous números decimais, e na variable op encóntrase un dos catro operadores aritméticos (+, –, x, /), realizar a operación aritmética (a op b), utilizando sentenzas if-else.

```
#include <iostream>
using namespace std;
int main ()
{
    // Decláranse as variables que utilizaremos
    float a, b;
    char op;
    /* Pídense os operandos x e y, e o operador */
```

```
        cout << "Introducir o primeiro operando.“ <<
            endl;
        cout << "a: ";cin >> a;
        cout << "\nIntroducir o operador (+, –, x, /).“
            << endl;
        cout << "op: ";cin >> op;
        cout << "\nIntroducir o segundo operando.” <<
            endl;
        cout << "b: ";cin >> b;
        /* Realízase a operación aritmética, usando só
            sentenzas if-else */
        if (op == ‘+’)
            cout << "\na + b = " << a << " + " << b << "
                = " << a + b << endl;
            else
             if (op == ‘-’)
                cout << "\na - b = " << a << " – " << b
                    << " = " << a – b << endl;
             else
             if (op == ‘x’)
                cout << "\na x b = " << a << " x " << b
                    << " = " << a * b << endl;
             else
             if (op == ‘/’)
                 cout << "a/b = " << a << "/" << b << " =
                     " << a/b << endl;
             else
             cout<<”Tecleaches un operador inválido”;
        return 0;
    } //fin main()
```

¿Por que ten que ir o operador entre comiñas simples nas comparacións?

Porque estamos comparando o contido da variable op, co código ASCII que corresponde ao carácter que representa ao operador.

Por exemplo, se escribimos if(op == x), entón compárase o contido da variable op, co contido da variable x, supoñendo que esta última estivese definida.

¿Cantas comparacións require a suma e a multiplicación?

A suma soamente unha, e a multiplicación tres, as mesmas que a división.

```
    #include <iostream>
    using namespace std;
    int main () {
        //Decláranse as variables que utilizaremos
        float a, b;
        char op;
        /* Pídense os operandos x e y, e o operador */
```

```
        cout << "Introducir o primeiro operando." <<
            endl;
        cout << "a: ";cin >> a;
        cout << "Introducir o operador (+, -, x, /)."<<
            endl;
        cout << "op: ";cin >> op;
        cout << "Introducir o segundo operando."<< endl;
        cout << "b: ";cin >> b;
        /* Realízase a operación aritmética, usando
            soamente sentenzas if-else */
        switch (op)
        {
            case '+':
                cout << "a + b = " << a << " + " << b <<
                    " = " << a + b << endl;
                break;
            case '-':
                cout << "a-b=" <<a << "-" <<b<<"=" <<a-b
                    << endl;
                break;
            case 'x':
                cout << "a x b = " << a << " x " << b <<
                    " = " << a * b << endl;
                break;
            case '/':
                cout << "a/b = " << a << "/" << b << " =
                    " << a/b << endl;
                break;
            default:
                cout <<“Non introduxiste un operador
                    valido" << endl;
        }
        system ("pause");
        return 0;
    } //fin main()
```

Se a, b e c son os lados dun triángulo, saber se é equilátero, isósceles ou escaleno.

```
    #include <iostream>
    using namespace std;
    int main ()
    {
        //Decláranse as variables que utilizaremos
        inta, b, c;
        /* Pídense os lados do triángulo */
        cout << "Introducir os lados a, b e c.“ << endl;
```

```
    cout << "a: ";cin >> a;
    cout << "b: ";cin >> b;
    cout << "c: ";cin >> c;
    /* Usando dúas sentenzas if-else e operadores
        lóxicos,sabemos si o triángulo é equilátero,
        isósceles ou escaleno.*/
    if ((a == b) && (b == c))//triángulo equilátero
        cout << "O triángulo éequilátero“ << endl;
    else
        if ((a == b) || (a == c) || (b == c))
          //triángulo isósceles
          cout << "O triángulo éisósceles“ << endl;
         else
          cout << "O triángulo é escaleno“ << endl;
    system(“pause”);
    return 0;
}
```

Resolver o problema anterior para tódolos casos, sen usar operadores lóxicos.

```
#include <iostream>
using namespace std;
int main ()
{
    //Decláranse as variables que utilizaremos
    int a, b, c;
    /* Pídense os lados del triángulo */
    cout << "Introducir los lados a, b y c.“ << endl;
    cout << "a: ";cin >> a;
    cout << "b: ";cin >> b;
    cout << "c: ";cin >> c;
    /* Usando sentencias if-else, sabemos se o
        triángulo é equilátero, isósceles ou
        escaleno.*/
    if (a == b)
        if (b == c)
            cout <<"Otriánguloéequilátero“ ;
        else //b!=c
            cout <<"Otriánguloéisósceles“ << endl;
        else//a es distinto de b.
        if (b == c)
            cout <<"Otriánguloéisósceles“ << endl;
        else
        if (a == c)
            cout <<"Otriánguloéisósceles” << endl;
        else
            cout <<"Otriánguloé escaleno“ << endl;
    system(“pause”);
```

```
    return 0;
}
```

Volver realizar oexercicio anterior, comparando soamente por diferente, esen utilizar operadores lóxicos. Recordar que a desigualdade noné transitiva, é dicir, se a ≠ b, y b ≠c, non implica que a ≠c.

```
#include <iostream>
using namespace std;
int main ()
{
    //Decláranse as variables que utilizaremos
    int a, b, c;
    /* Pídense os lados do triángulo */
    cout << "Introducir os lados a, b e c.“ << endl;
    cout << "a: "; cin >> a;
    cout << "b: "; cin >> b;
    cout << "c: "; cin >> c;
    /* Usando sentenzas if-else, e comparando
        soamentepor desigual, sabemos se o triángulo
        é equilátero, isósceles ou escaleno.*/
    if (a != b)
        if (b != c)
         if (a != c)
             cout << "Otriánguloé escaleno“ << endl;
         else //a==c
             cout << "Otriánguloéisósceles“ << endl;
         else//b==c
             cout << "Otriánguloéisósceles“ << endl;
         else//a==b
         if (b != c)
             cout << "Otriánguloéisósceles“ << endl;
         else //a==b y b==c
             cout << "Otriánguloéequilátero“ << endl;
    system(“pause”);
    return 0;
}
```

Implementar o proceso de división usando un `while`, calculando con cantidades enteiras. Hase de proporcionar o cociente e o resto.

```
#include <iostream>
using namespace std;
int main ()
{
//Decláranse as variables que utilizaremos
```

```
int dividendo, divisor, cociente = 0;
//É importante inicializar a 0 o cociente.
/* Pídense os operandos */
     cout << "Introducir o dividendo eo divisor.“ <<
          endl;
     cout << "Dividendo: ";
     cin >> dividendo;
     cout << "Divisor: ";
     cin >> divisor;
     /*Utilizando un while realizamos a división*/
     while (dividendo >= divisor)
     {
      cociente++;
      dividendo -= divisor;//El resto queda en el
           dividendo
     }
     cout << "O cociente é:\t" << cociente << endl;
     cout << "Eo resto é:\t" << dividendo << endl;
     return 0;
}
```

Incorporar un do-whileao código del problema 7, de modo que o programa pregunte se se quere realizar outra división. O programa finalizará cando a resposta sexa non.

```
#include <iostream>
using namespace std;
int main ()
{
     //Decláranseas variables que utilizaremos
     int dividendo, divisor, cociente;
     char resposta;
     do {
      // Pídense os valores dividendo e divisor
      cout << "Introducir o dividendo e o
           divisor."<<endl;
      cout << "Dividendo: ";
      cin >> dividendo;
      cout << "Divisor: ";
      cin >> divisor;
      cociente = 0;//Hase de inicializar para cada
           división.
      //Utilizando un while realizamos a división
      while (dividendo >= divisor)
      {
           cociente++;
           dividendo -= divisor;
           //O resto queda no dividendo
      }
```

```
        cout << "O cociente é:\t" << cociente << endl;
        cout << "eo resto é:\t" << dividendo << endl;
        cout << endl << endl << "Quere realizar outra
             división? (s/n):";
        cin >>resposta;
      } while ((resposta == ‘s’) || (resposta ==
          ‘S’));
      return 0;
}
```

Implementar oproduto de dous números enteiros mediante un `for`.

```
#include <iostream>
using namespace std;
int main ()
{
      //Decláranse as variables que utilizaremos
      int mult, multdor, i, produto = 0;
      /* Pídense os operandos */
      cout << "Introducir o multiplicando eo
      multiplicador.“ << endl;
      cout << "Multiplicando: ";
      cin >> mult;
      cout << "Multiplicador: ";
      cin >> multdor;
      // Utilizando un for realizamos o produto
      for (i = 0; i < multdor; i++)
            produto += mult;
      cout << "Oprodutoé:" <<produto<< endl;
      return 0;
}
```

ANEXO: ACTIVIDADES PROPOSTAS

1. Escribir un programa en C++ que calcule a lonxitude e a área dunha circunferencia
   Lonxitude da circunferencia = 2 * PI * raio.
   Área da circunferencia = PI * raio^2
2. Escribir un programa en C++ que calcule a velocidade dun proxectil que recorre un espazo determinado nun tempo. Expresar o resultado en metros/segundo.
   Velocidade = espazo/tempo
3. Escribir un programa en C++ que calcule o volume dunha esfera
   Volume da esfera = 4/3 * PI * radio^3
4. Escribir un programa en C++ que calcule as raíces reais dunha ecuación de 2º grao
5. Escribir un programa en C++ que calcule o número de horas, minutos e segundos que hai en X segundos.
6. Escribir un programa en C++ que calcule o capital producido por un capital de X Euros, o cabo dun ano depositado a un xuro do 2%.
7. Escribir un programa en C++ que calcula a seguinte expresión trigonométrica.
   (sen x * cos x)/(tan x)
8. Escribir un programa en C++ que calcule o equivalente en pe´s dunha lonxitude en metros.
   1 metro ----------39.27 polgadas
   12 polgadas --------1 pé
9. Escribir un programa en C++ que calcule a área dun rectángulo a partir das súas coordenadas (X1,Y1) Y (X2,Y2)
10. Un foguete lánzase verticalmente cunha velocidade de V m/s. Calcular a velocidade ao cabo de t segundos mediante un programa en C++
    Velocidade instantánea = (velocidade inicial) - (aceleración da gravidade * tempo)
11. Escribir un programa en C++ que dado un número do 1 a 7 escriba o correspondente nome do día da semana.
12. Escribir un programa en C++ que lea desde teclado o importe bruto dunha factura e determine o importe neto segundo os seguintes criterios.
    - Importe bruto menor de 20.000 -> sen desconto
    - Importe bruto maior de 20.000 -> 15% de desconto
13. Escribir un programa en C++ que unha vez lida unha hora en formato horas, minutos, segundos, indique cal será o tempo dentro de x segundos
14. Escribir un programa en C++ que visualice en pantalla os números múltiplos de x comprendidos entre 1 y 1000, sendo x un número enteiro entre 1 y 10.
15. Sabemos que sumando o primeiro impar, obtense o primeiro cubo; sumando os dos seguintes impares, obtense o segundo cubo; sumando os tres seguintes impares, obtense o terceiro cubo, . . . E

dicir: $1^3 = 1$, $2^3 = 3 + 5 = 8$, $3^3 = 7 + 9 + 11 = 27$, . . .
Escribe un programa que calcule e escriba tantos cubos como o usuario desexe.

16. Realiza un programa que reciba un ano como entrada e indique si é bisesto ou non. Nota: Son bisestos tódolos anos divisibles por 4, excepto os divisibles por 100 que non son divisibles por 400. Ex. 2000 é bisesto, pero 1900 non foi bisesto.
17. Realiza un programa que busque e mostre por pantalla tódolos números de tres cifras para os que a suma do cubo de súas cifras iguale ao propio número. Por exemplo, $153=1^3+5^3+3^3$.
18. En matemáticas, unha serie de Taylor dunha función f(x) infinitamente derivable (real o complexa) definida nun intervalo aberto (a-r, a+r) defínese como a seguinte suma:

$$f(x) = \sum_{n=0}^{\infty} \frac{f_{(a)}^{n}}{n!}(x-a)^n$$

onde, n! es o factorial de n e $f_{(a)}^{n}$ indica a n-ésima derivada de f no punto a.

a) Realizar un programa para calcular a función seno(x) mediante unha serie de Taylor

$$seno(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}x^{2n+1}$$

b) Realizar un programa para calcular a función cos(x) mediante unha serie de Taylor

$$cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!}x^{2n}$$

c) Realizar un programa para calcular a función ln(1+x) mediante unha serie de Taylor

$$ln(1+x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}x^n \;\; si\ |x| < 1$$

19. Sen copiar estes programas no ordenador, cal é a saída dos seguintes programas en C++?

```
#include <iostream>
using namespace std;
int main()
{
int celsius;
for (celsius = 0; celsius
<= 100; celsius++);
   cout<<”Celsius:
“<<celsius<<”Fahrenheit:
“<<
(celsius * 9) / 5 +
32<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main(){
int debe;
cout<<"Introduza cantidade
que se debe: ";
cin>>debe;
if (debe = 0)
cout<<"Non se debe
nada."<<endl;
else
cout<<"Débense
"<<debe<<”euros”;
return 0;}
```

AVALIACIÓN DA UD

A avaliación desta UD enmárcase nun proceso de avaliación continua. O alumnado terá que realizar todas as actividades propostas, e estas han de ser entregadas en forma e prazo. A asistencia, participación nas clases prácticas e entregas das actividades correctas, contará para a nota final cun valor de ata 1/3 da nota final

Periodicamente realizaranse en calquera momento durante as clases controis de prácticas, é dicir, exercicios prácticos que o alumnado ten que resolver individualmente sen axuda de medios electrónicos

BIBLIOGRAFÍA BÁSICA

• Joyanes Aguilar, Luis. Metodología de la programación. Mc Graw Hill
• Joyanes Aguilar, Luis, Rodríguez Baena, L. e Fernández Azuela, Matilde. Fundamentos de programación, libro de problemas. Mc graw Hill.
[1] Programación en C Byron S. Gottfried. Ed Mc Graw Hill. Serie Schaum.
[2] Practical C Programming Steve Oualline. Ed. O'Reilly & Associates, Inc.
[3] Data Structures Using C A. M. Tenenbaum, Y. Langsam & M. J. Augenstein. Ed. Prentice-Hall
[4] El lenguaje de programación C B. W. Kernighan, D. N. Ritchie. Ed Prentice-Hall

unidadesdidácticas
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Unha colección orientada a editar materiais docentes de calidade e pensada para apoiar o traballo do profesorado e do alumnado de todas as materias e titulacións da universidade

Impreso en papel 100% reciclado e libre de cloro

USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
SERVIZO DE NORMALIZACIÓN LINGÜÍSTICA

9 788498 879346