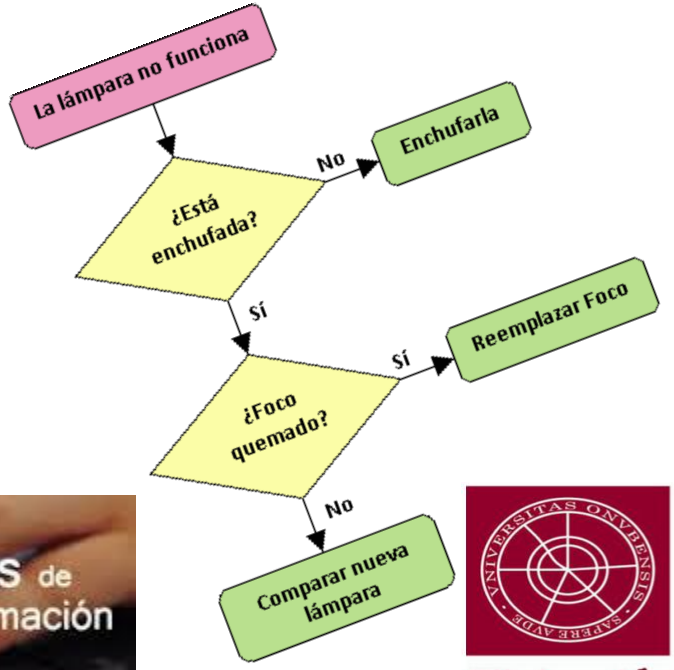
Tema 2.

ALGORITMOS. Tipos de Datos...

Fundamentos de Programación
Grado en Ingeniería Informática



Profesores:
José Manuel Martín Ramos
Francisco Roche Beltrán.

ETSI ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

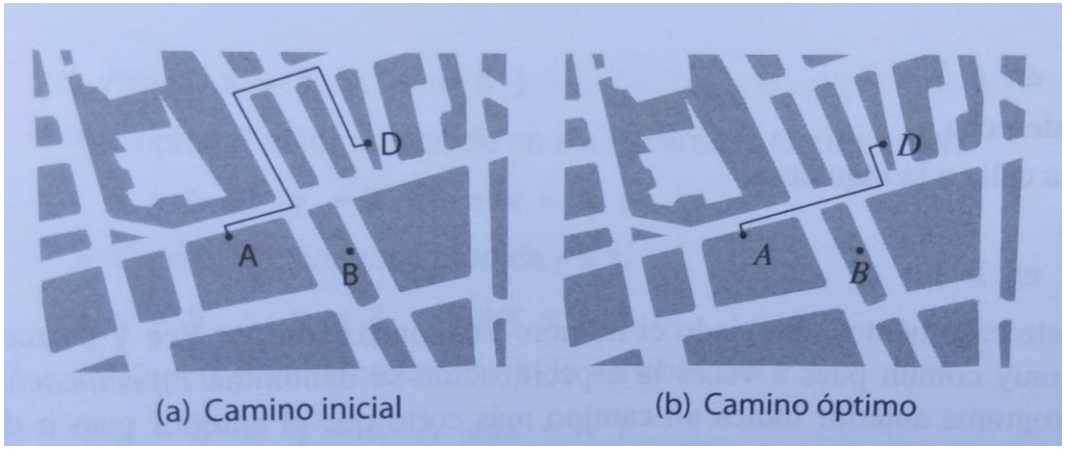
DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN

Universidad de Huelva

ÍNDICE

1. Concepto de algoritmo. Estructura general.

Algoritmo es una sucesión ordenada de acciones que se deben de realizar para conseguir la solución correcta de un problema en un espacio de tiempo finito.



Programa es un algoritmo escrito en un lenguaje de programación.

La estructura general de un programa en el lenguaje C++ es:

Sin orientación a objetos	Con orientación a objetos
<inclusión de bibliotecas> <programa principal>	<inclusión de bibliotecas> <declaración de clase> <programa principal>

`#include <iostream>`
`using namespace std;`
 	
`int main () {`
`  int a;`
`  a = 1;`
`  cout << a;`
`  return 0;`
`}`
	`#include <iostream>`
	`using namespace std;`
	`class fraccion{`
	`…`
	`};`
	`int main () {`
	`  int a;`
	`  a = 1;`
	`  cout << a;`
	`  return 0;`
	`}`

2. Palabras reservadas. Identificadores…

- Palabras reservadas.

```
class, for, while, public, delete, new...
```

- Identificador.

Identificadores válidos	Identificadores no válidos
• A1876	• Cantidad total
• X_W_W_1	• 2numeros
• Cinco_numeros	• for
• TASA	• %cambiodetasa

El lenguaje C++ es sensible a las mayúsculas: Saldo, saldo y SALDO definen identificadores distintos.

- Constantes son zonas de memoria cuyo valor **NO** se puede modificar durante la ejecución del programa.

```
const    int    k = 12;
```

```
#include <iostream>
using namespace std;
int main() {
    int radio;
    const float pi=3.14159;
    float longitud;
    ...
    longitud = 2 * pi * radio;
    superficie = pi * radio * radio;
    ...
    return 0;
}
```

```
#include <iostream>
using namespace std;
#define pi 3.14159
int main() {
    int radio;
    float longitud;
    ...
    longitud = 2 * pi * radio;
    superficie = pi * radio * radio;
    ...
    return 0;
}
```

```
#define    INF          -30
#define    letra        'A'
#define    MENSAJE       "Introduzca    su  edad:"
```

"\n"	Salto de línea	cout <<"\n";
"\t"	Tabulador	cout <<"\t";

- **Variables** son zonas de memoria cuyo valor se puede modificar durante la ejecución del programa.

Sintaxis para la declaración de variables:

```
Nombre_de_tipo   Nombre_1, Nombre_2...;
```

```
#include <iostream>
using namespace std;
int main() {
    int radio;
    int longitud=33;
    ...
    return 0;
}
```

- Comentarios:

```
#include <iostream>
using namespace std;
#define pi 3.14159
int main() {
    int radio;
    float longitud, superficie;
    ...
    //Calculo la longitud de la circunferencia
    longitud = 2 * pi * radio;
    /*Voy a calcular la superficie
    del círculo */
    superficie = pi * radio * radio;
    ...
    return 0;
}
```

3. Tipos de Datos: clasificación.

3.1. Tipo entero.

```
int a;
```

El número mayor y menor que puede representar depende del tamaño del espacio usado por el dato.

Si ocupa 2 bytes → Entre -32768 y +32767

Si ocupa 4 bytes → Entre -2147483648 al 2147483647

```
unsigned int a;
```
/*Entero sin signo, de 0 a 65535 si ocupa 2 bytes, o de 0 a 4294967295 si ocupa 4 bytes*/

```
long z;
```
// Entero largo, doble del tamaño que un entero

Operaciones con variables de tipo entero:

Suma	+	Resta	-	Multiplicación	*
Cociente	/	Resto (Módulo)	%		

3.Tipos de Datos: clasificación.

3.2. Tipo real.

```
float a;
```

Desde $-3.4 * 10^{38}$ hasta $-3.4 * 10^{-38}$ y

desde $3.4 * 10^{-38}$ hasta $3.4 * 10^{38}$.

```
double  r;
```

Desde $-1.79 * 10^{308}$ hasta $-1.79 * 10^{-308}$ y

Desde $1.79 * 10^{-308}$ hasta $1.79 * 10^{308}$.

Operaciones con variables de tipo real:

Suma	+	Resta	-
Multip.	*	División	/

Los números reales utilizan . y no la , para señalar los decimales, por ejemplo ~~3,14159~~ incorrecto 3.14159 correcto

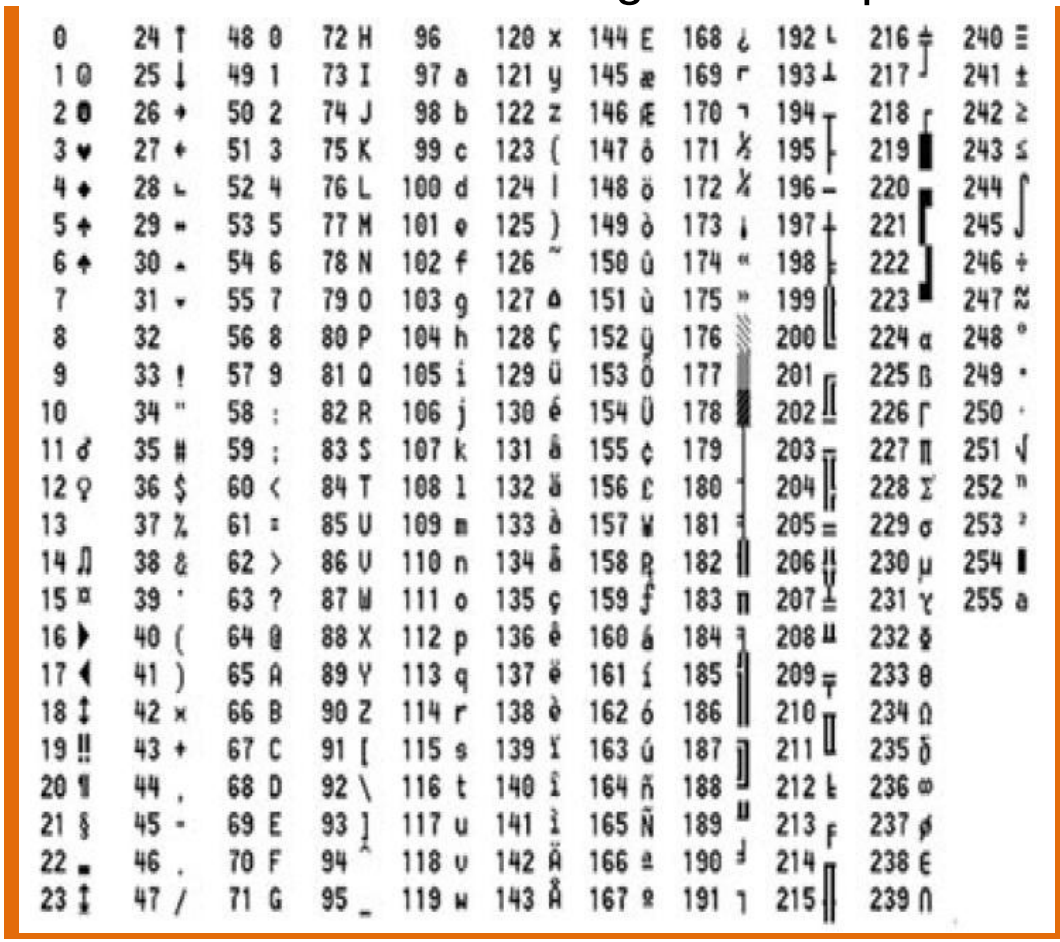
En notación científica sería. 1.456e10 para expresar 14 560 000 000.

O bien: 0.000000034 debería de ponerse 3.4e-8

3.Tipos de Datos: clasificación.

3.3 Tipo carácter.

Se almacena el código ASCII que coresponde al carácter.



0	24 ↑	48 0	72 H	96 `	120 x	144 É	168 ¿	192 └	216 ╪	240 ≡
1 ☺	25 ↓	49 1	73 I	97 a	121 y	145 æ	169 ⌐	193 ┴	217 ┘	241 ±
2 ☻	26 →	50 2	74 J	98 b	122 z	146 Æ	170 ¬	194 ┬	218 ┌	242 ≥
3 ♥	27 ←	51 3	75 K	99 c	123 {	147 ô	171 ½	195 ├	219 ▉	243 ≤
4 ♦	28 ∟	52 4	76 L	100 d	124 |	148 ö	172 ¼	196 ─	220 ▄	244 ⌠
5 ♣	29 ↔	53 5	77 M	101 e	125 }	149 ò	173 ¡	197 ┼	221 ▌	245 ⌡
6 ♠	30 ▲	54 6	78 N	102 f	126 ~	150 û	174 «	198 ╞	222 ▐	246 ÷
7	31 ▼	55 7	79 O	103 g	127 ⌂	151 ù	175 »	199 ╟	223 ▀	247 ≈
8	32	56 8	80 P	104 h	128 Ç	152 ÿ	176 ░	200 ╚	224 α	248 °
9	33 !	57 9	81 Q	105 i	129 ü	153 Ö	177 ▒	201 ╔	225 ß	249 ∙
10	34 "	58 :	82 R	106 j	130 é	154 Ü	178 ▓	202 ╩	226 Γ	250 ·
11 ♂	35 #	59 ;	83 S	107 k	131 â	155 ¢	179 |	203 ╦	227 π	251 √
12 ♀	36 $	60 <	84 T	108 l	132 ä	156 £	180 ┤	204 ╠	228 Σ	252 ⁿ
13	37 %	61 =	85 U	109 m	133 à	157 ¥	181 ╡	205 ═	229 σ	253 ²
14 ♫	38 &	62 >	86 V	110 n	134 å	158 ₧	182 ╢	206 ╬	230 µ	254 ■
15 ☼	39 '	63 ?	87 W	111 o	135 ç	159 ƒ	183 ╖	207 ╧	231 τ	255
16 ►	40 (	64 @	88 X	112 p	136 ê	160 á	184 ╕	208 ╨	232 Φ	
17 ◄	41)	65 A	89 Y	113 q	137 ë	161 í	185 ╣	209 ╤	233 Θ	
18 ↕	42 *	66 B	90 Z	114 r	138 è	162 ó	186 ║	210 ╥	234 Ω	
19 ‼	43 +	67 C	91 [	115 s	139 ï	163 ú	187 ╗	211 ╙	235 δ	
20 ¶	44 ,	68 D	92 \	116 t	140 î	164 ñ	188 ╝	212 ╘	236 ∞	
21 §	45 -	69 E	93]	117 u	141 ì	165 Ñ	189 ╜	213 ╒	237 ø	
22 ▬	46 .	70 F	94 ^	118 v	142 Ä	166 ª	190 ╛	214 ╓	238 ε	
23 ↨	47 /	71 G	95 _	119 w	143 Å	167 º	191 ┐	215 ╫	239 ∩	

```
char a;
char a='0';
char a=48;
```

3.Tipos de Datos: clasificación.

3.4. Tipo Enumerado

```
enum  nombre_del_tipo {
        definición de nombres de constantes};
```

```
enum dia {
lunes, martes, miercoles, jueves, viernes,sabado, domingo};

dia hoy; hoy = jueves;
```

3.5 Tipo lógico (booleano).

Las variables o constantes pueden tomar cierto (true) o falso (false).

```
bool a;
a = true;
a = false;
```

3.Tipos de Datos: clasificación.

3.6. Tipos de datos estructurados.

Construidos a partir de otros tipos de datos de tipo simple con las relaciones existentes entre ellos.

```
int      lista[40];
char     palabra[20];
char     s[20]= ”Francisco”;
```

	Homogénea	**Heterogénea**
Estática	Tablas	Registros
Dinámica	Tablas dinámicas	Memoria Dinámica

Los tipos de datos estructurados dinámicos serán motivo de estudio en la asignatura de Estructuras de Datos I del 2º cuatrimestre de primer curso.

4. Operaciones básicas de entrada-salida.

Para utilizar las instrucciones de lectura de datos desde teclado y la salida de datos por pantalla hay que añadir al comienzo del programa:

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

4.1 Lectura

```
cin >> identificador_una_variable;
```

```
cin >> var1;

cin >> var1 >> var2;


cin >> var1
    >> var2;
```

```
char  c;

cin >> c;


char  s[80];

cin >>   s;
```

4. Operaciones básicas de entrada-salida.

4.2 Escritura.

```
cout <<  expresión;
```

```
int main () {
   int  a, b;
   …
   cout <<  a;
   cout << a * b;
   cout << "El valor obtenido es " << a;
   cout << "El valor obtenido es "
        << a;
   cout << "\n";
   cout << "La cantidad obtenida es :\n" << a;
   return 0;
}
```

5. Operadores.

5.1. Operador de asignación.

Identificador_variable = expresión;

```
int a;
a = 10;
a = 3 + 9;
a = a + 1;
```

```
#include <iostream>
using namespace std;
#define pi 3.14159
int main() {
    int radio;
    float longitud, superficie;
    ...
    //Calculo la longitud de la circunferencia
    longitud = 2 * pi * radio;
    /*Voy a calcular la superficie
    del círculo */
    superficie = pi * radio * radio;
    ...
    return 0;
}
```

5. Operadores.

5.2. Operadores aritméticos.

+ , - , * , /

Los operandos pueden ser de tipo entero o reales. Si en la división los dos operandos son enteros el resultado es un entero que corresponde con el cociente, perdiéndose el resto.

```
int valor;
valor = 5/3;
cout << valor; //Mostrará por pantalla el valor 1.
```

%

Los operandos tienen que ser de tipo entero.

```
int valor;
valor = 5%3;
cout << valor; //Mostrará por pantalla el valor 2.
```

Sea una variable de tipo entero denominada x:

x++;	x = x + 1;
++x,	x = x + 1;
x--;	x = x - 1;
--x;	x = x - 1;
x += 2;	x = x + 2;
x -= 3;	x = x - 3;

```
x *= 3;          x = x * 3;

x /= 2;          x = x / 2;

y = x++;         y = x;
                 x = x + 1;

y = ++x;         x = x + 1;
                 y = x;

int uno, dos;    int uno, dos;
uno=dos=1;       uno = 1;
                 dos = 1;
```

5. Operadores.

5.3. Operadores relacionales.

= =	(igual)
! =	(distinto)
< > < = > =	

Los operandos pueden ser de tipo entero o real y el resultado será true si se cumple la relación o bien false si no se cumple.

```
bool   p;
float x=15, y=18;

p = (x == y);
```

5. Operadores.

5.4. Operadores lógicos.

&& que es AND.

|| que es OR.

! que es NOT

Operador lógico AND

x	y	resultado
true	true	true
true	false	false
false	true	false
false	false	false

Operador lógico OR

x	y	resultado
true	true	true
true	false	true
false	true	true
false	false	false

6. Expresiones y orden de precedencia.

!, ++, --, - (unario), new, delete	derecha a izquierda
*, /, %	izquierda a derecha
+, - (binario)	izquierda a derecha
<, <=, >,>=	izquierda a derecha
==, !=	izquierda a derecha
&&	izquierda a derecha
||	izquierda a derecha

```
long a;
unsigned char b;
int c, f;
float d;
f = a + b * c * d;
```

Función cast (conversión explícita de tipos)

```
int n=7;
float a;
a =  n / 2;              //¿Cuanto vale a?
a = (float) n / 2;       //¿Cuanto vale a?
```

7. Funciones genéricas y métodos.

7.1. Funciones genéricas.

Una función es una colección de declaraciones de variables y de sentencias.

```
int main () {
    int a, b, c;
    a = 1;
    b = 3;
    c = a + b;
    return 0;
}
```

```
int suma () {
    int dato1, dato2, dato3;
    dato1 = 1;
    dato2 = 2;
    dato3 = dato1 + dato2;
    return dato3;
}
```

Sintaxis de una función:

```
tipodevuelto   nombre ( ) {
…
}
```

```
void multiplicar () {
    float a, b, c;
    a = 4.2;
    b = 2.3e-2;
    c = a * b;
    cout << c;
}

int main () {
    ...
    multiplicar ();
    ...
    return 0;
}
```

```
float multiplicar () {
    float a, b, c;
    a = 4.2;
    b = 2.3e-2;
    c = a * b;
    return c;
}

int main () {
    float valor;
    valor = multiplicar ();
    cout << valor;
    return 0;
}
```

7.2. Métodos.

Con orientación a objetos

<inclusión de bibliotecas>
<declaración de clase>
<programa principal>

```
#include <iostream>
using namespace std;
class fraccion{
…
};
int main () {
  int a;
  a = 1;
  cout << a;
  return 0;
 }
```

```
class fraccion {
  Atributos y funciones privadas
de la clase

public:

    Funciones públicas de la clase
};

//observe que acaba con;
```

```
class      miclase  {
  int a;
public:
  void poner();
  int leer ();
};
void miclase::poner(){
  a = 10;
}
int miclase::leer() {
  return a;
}
```

```
int main()  {
  int valor1,valor2;
  miclase ob1,ob2;
  ob1.a = 10; //Error
  ob1.poner();
  ob2.poner();
  valor1 = ob1.leer();
  valor2 = ob2.leer();
  cout << valor1;
  cout << valor2;
  return 0;
}
```

Sintaxis de un método:

tipodevuelto **nombre_clase::**nombre_método ( ) {
… }

Método es una función definida dentro de una class. (poner, leer).
Atributo es una variable definida dentro de una class. (a)

Operador de asignación entre objetos.

```
class     miclase {

  int a,b;

public:

  void poner();

};
void miclase::poner () {
  a = 5;
  b = 6;
}
```

```
int main()  {

  miclase ob1, ob2;

  ob1.poner();

  ob2 = ob1;     //ATENCION

  return 0;
}
```