

Programación 1

Tema 3

Información, datos, operaciones y expresiones



Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza





Índice

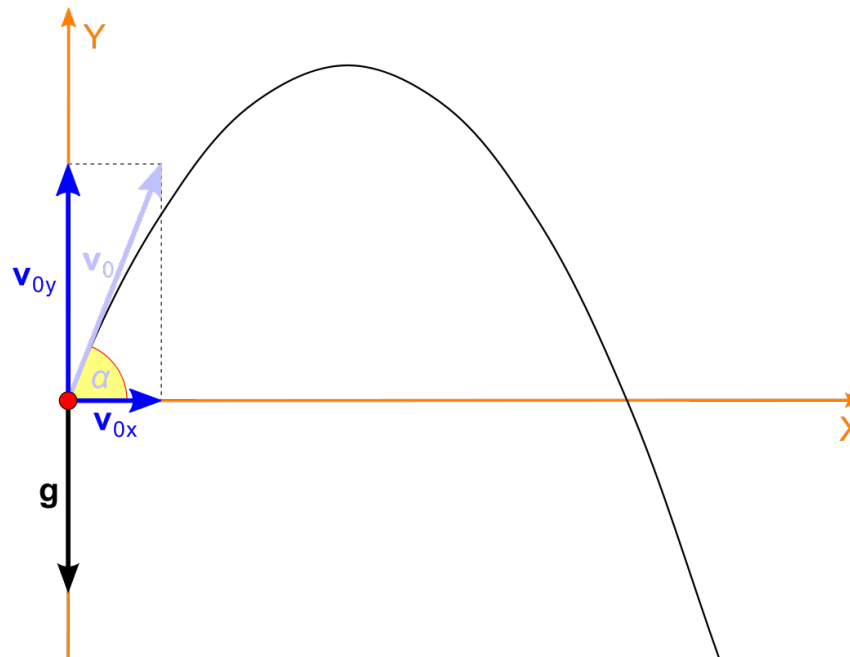
- ❑ Datos y tipos de datos
- ❑ Datos primitivos en C++
- ❑ Expresiones e instrucción de asignación

Datos y tipos de datos

- ❑ Problema → información → abstracción → datos
- ❑ Cada dato tiene un **valor**
- ❑ Con los datos se realizan cálculos y operaciones

Ejemplo

- ¿Qué información hace falta para resolver el problema de la trayectoria que sigue un proyectil?



Ejemplo

Lanzamiento de un proyectil

□ Información relevante

- Velocidad inicial v_0
- Ángulo de tiro α
- Posición inicial \vec{r}
- Aceleración de la gravedad g
- Coeficiente de rozamiento μ

□ Información no relevante (pero de tipos de datos distintos a \mathbb{R})

- Número de perdigones n
- ¿Es de día o de noche?
- Nombre del fabricante de los perdigones
- Datos personales de la persona que dispara

Datos en C++

- Tipos primitivos de datos
 - No derivan de otros tipos de datos
 - Dominio finito de valores
 - Codificación binaria definida
 - Sintaxis para representar sus valores
 - Operaciones predefinidas
- Tipos estructurados

Tipos primitivos en C++

- Enteros
 - short, **int**, long, long long
 - unsigned short, **unsigned int**, unsigned long, unsigned long long
- Reales
 - float, **double**, long double
- Booleanos
 - **bool**
- Caracteres
 - **char**

Tipos enteros

- Dominio de valores (GCC y MinGW)
 - Subconjunto de \mathbb{Z}
 - short int -32768..32767
 - **int** **$-2 \times 10^9 .. 2 \times 10^9$**
 - long int $-2 \times 10^9 .. 2 \times 10^9$
 - long long int $-9 \times 10^{18} .. 9 \times 10^{18}$
- Representación externa en C++
 - 0 1 -1 6 2541 ...
- Codificación
 - Complemento a dos (16, 32 o 64 bits)

Tipos enteros

- Dominio de valores (GCC y MinGW)
 - Subconjunto de \mathbb{Z}
 - `short int` $-32768..32767$
 - `int` $-2 \times 10^9 .. 2 \times 10^9$
 - `long int` $-2 \times 10^9 .. 2 \times 10^9$
 - `long long int` $-9 \times 10^{18} .. 9 \times 10^{18}$
- Representación externa en C++
 - `0` `1` `-1` `6` `2541` ...
- Codificación
 - Complemento a dos (16, 32 o 64 bits)

Tipos enteros

- Dominio de valores (GCC y MinGW)
 - Subconjunto de \mathbb{N}
 - unsigned short int 0..65535
 - **unsigned int** **$0..4 \times 10^9$**
 - unsigned long int $0..4 \times 10^9$
 - unsigned long long int $0..18 \times 10^{18}$
- Representación externa en C++
 - 0 1 6 2541 ...
- Codificación
 - Binaria (16, 32 o 64 bits)

Tipos enteros

□ Dominio de valores (GCC y MinGW)

■ Subconjunto de \mathbb{N}

- `unsigned short int` 0..65535
- `unsigned int` **0..4×10⁹**
- `unsigned long int` 0..4×10⁹
- `unsigned long long int` 0..18×10¹⁸

□ Representación externa en C++

- 0 1 6 2541 ...

□ Codificación

- Binaria (16, 32 o 64 bits)

Tipos reales

- Dominio de valores (GCC y MinGW)
 - Subconjunto de \mathbb{R}
 - float $-3.40282 \times 10^{38} \dots +3.40282 \times 10^{38}$
 - double **$-1.79769313 \times 10^{308} \dots +1.79769313 \times 10^{308}$**
 - long double $-1.1897315 \times 10^{4932} \dots +1.1897315 \times 10^{4932}$
- Representación externa en C++
 - 0.0 0.5 -1.75 3.14159265358979323846
6.022e23 -1.602e-19
- Codificación
 - IEEE 754 (32, 64 o 96 bits)

Booleanos

- ❑ **bool**
- ❑ Dominio de valores
 - {falso, cierto}
- ❑ Representación externa en C++
 - **false true**
- ❑ Codificación
 - 8 bits

Caracteres

- **char**
- **Dominio de valores**
 - 96 caracteres del alfabeto inglés
 - Letras
 - Dígitos
 - Signos de puntuación
 - Otros símbolos
 - 32 *caracteres* de control
 - 128 caracteres dependientes de la codificación

	0	@	P	`	p
!	1	A	Q	a	q
"	2	B	R	b	r
#	3	C	S	c	s
\$	4	D	T	d	t
%	5	E	U	e	u
&	6	F	V	f	v
'	7	G	W	g	w
(8	H	X	h	x
)	9	I	Y	i	y
*	:	J	Z	j	z
+	;	K	[k	{
,	<	L	\	l	
-	=	M]	m	}
.	>	N	^	n	~
/	?	O	_	o	

Caracteres

□ Representación externa en C++

- 'a' 'A' 'b' 'B' 'z' 'Z'
- '0' '1' '2' '3' '4' '5' '6' '7'
'8' '9'
- '+' '-' '*' '/' '<' '=' '>'
- '(' ')' '[' ']' '{' '}'
- '#' '\$' '%' '&' ',' '.' ':' ';'
'!' '?' '@' '^' '_' '`' '|' '~'
- '"' '\'

Operaciones (datos primitivos)

- Unitarias (enteros y reales)
 - +, -
- Aritméticas (enteros y reales)
 - +, -, *, /, %
- Lógicas (*booleanos*)
 - !, &&, ||
- Relacionales (enteros, reales, caracteres, *booleanos*, ...)
 - ==, !=
 - >, >=, <, <=

Datos constantes y variables

- Constantes literales
 - 0, 25, -8, 3.14159, **true**, **false**, 'a', 'Z', "Universidad de Zaragoza"
- Constantes simbólicas
 - **const int** MAXIMO = 1000;
 - **const int** ANCHO = 9;
 - **const double** PI = 3.141592653589793;
- Variables
 - Variables locales
 - Parámetros de una función



Variables

- Datos cuyo **valor** puede variar entre ejecuciones
 - O incluso en la misma ejecución
- Siempre tienen un valor asociado
- En C++ tienen asociado un tipo no modificable



Variables

00000101	@1800
10110110	@1801
01000110	@1802
10101110	@1803
10101000	@1804
00110001	@1805
01101011	@1806
00001011	@1807
01110001	@1808
10101100	@1809
10011011	@1810
10001111	@1811
01110100	@1812



Variables

```
int a;
```

a

00000101	@1800
10110110	@1801
01000110	@1802
10101110	@1803
10101000	@1804
00110001	@1805
01101011	@1806
00001011	@1807
01110001	@1808
10101100	@1809
10011011	@1810
10001111	@1811
01110100	@1812



Variables

```
int a;  
int b = 3;
```

a	00000101	@1800
	10110110	@1801
	01000110	@1802
	10101110	@1803
b	00000000	@1804
	00000000	@1805
	00000000	@1806
	00000011	@1807
	01110001	@1808
	10101100	@1809
	10011011	@1810
	10001111	@1811
	01110100	@1812

Variables

```
int a;  
int b = 3;  
char c1;
```

a	00000101	@1800
	10110110	@1801
	01000110	@1802
	10101110	@1803
b	00000000	@1804
	00000000	@1805
	00000000	@1806
	00000011	@1807
c1	01110001	@1808
	10101100	@1809
	10011011	@1810
	10001111	@1811
	01110100	@1812

Variables

```
int a;  
int b = 3;  
char c1;  
char c2 = 'A';
```

a	00000101	@1800
	10110110	@1801
	01000110	@1802
	10101110	@1803
b	00000000	@1804
	00000000	@1805
	00000000	@1806
	00000011	@1807
c1	01110001	@1808
c2	01000001	@1809
	10011011	@1810
	10001111	@1811
	01110100	@1812

Variables

```
int a;  
int b = 3;  
char c1;  
char c2 = 'A';  
bool d = (b == 8);
```

a	00000101	@1800
	10110110	@1801
	01000110	@1802
	10101110	@1803
b	00000000	@1804
	00000000	@1805
	00000000	@1806
	00000011	@1807
c1	01110001	@1808
c2	01000001	@1809
d	00000000	@1810
	10001111	@1811
	01110100	@1812

Variables

```
int a;  
int b = 3;  
char c1;  
char c2 = 'A';  
bool d = (b == 8);
```

a	?
b	3
c1	?
c2	'A'
d	false

Declaración de variables

- Datos de tipos primitivos
 - `int i, j, k;`
 - `unsigned m, n;`
 - `char c1, c2;`
 - `bool b;`
 - `double r1, r2, r3;`

Declaración de variables

□ Datos de tipos primitivos

- `int i = 3,
j = 0,
k = 100;`
- `char c1 = 'h',
c2 = 'Y';`
- `bool b = true;`
- `double r1 = 0.0,
r2 = 1.5E6,
r3 = 0.56;`

Sintaxis de declaración de variables

- `<declaración> ::=`
 `<tipo> <declaraciónSimple>`
 `{ “,” <declaraciónSimple> } “;”`
- `<declaraciónSimple> ::=`
 `<nombre-variable> [“=” <expresión>]`

Sintaxis de declaración de variables

- `<declaración> ::=`
 `<tipo> <declaraciónSimple>`
 `{ “,” <declaraciónSimple> } “;”`
- `<declaraciónSimple> ::=`
 `<nombreVariable> [“=” <expresión>]`

Semántica de la declaración de variables

- Se reserva espacio en memoria para almacenar tantos datos del tipo especificado en la declaración como `<declaraciones-simples>` haya.
- Si la `<declaración-simple>` de una variable incluye inicialización, se evalúa la `<expresión>` de inicialización. La variable que se declara, pasa a tener ese valor inicial.
- Si la `<declaración-simple>` no incluye una expresión de inicialización, el valor de la variable declarada queda indefinido. Posteriormente en el programa habrá que darle valor inicial antes de consultar su valor.
- A partir de ese punto del programa, se puede trabajar con las variables declaradas (consultar sus valores o modificarlos).

Declaración de variables

- Datos de tipos primitivos
 - `int a;`
 - `int b = 1;`
 - `int n = 4 + 8;`
 - `char c = char(int('A') + 1);`
 - `bool b = (n == 12);`
 - `double r = sqrt(2.0);`

Ejemplo



Ejemplo

```
#include <iostream>
#include <iomanip>
using namespace std;
/*
 * Programa que escribe en la pantalla la cantidad que
 * equivale en euros a 2000 pesetas.
 */
int main() {
    const double PTAS_POR_EURO = 166.386;
    unsigned pesetas = 2000;
    double euros = pesetas / PTAS_POR_EURO;
    cout << fixed << setprecision(2) << euros << endl;
    return 0;
}
```

El mismo ejemplo, más general

```
#include <iostream>
#include <iomanip>
using namespace std;
/*
 * Programa que escribe en la pantalla la cantidad
 * equivalente en euros a una cantidad de dinero entera
 * expresada en pesetas solicitada previamente al usuario.
 */
int main() {
    const double PTAS_POR_EURO = 166.386;
    cout << "Escriba una cantidad en pesetas: ";
    unsigned pesetas;
    cin >> pesetas;
    double euros = pesetas / PTAS_POR_EURO;
    cout << fixed << setprecision(2) << euros << endl;
    return 0;
}
```



Índice

- Datos y tipos de datos
- Datos primitivos en C++
- **Expresiones e instrucción de asignación**

Sintaxis de la instrucción de *asignación*

```
<instrucción-asignación> ::=  
    <variable> “=” <expresión> “;”  
    | ...
```

Semántica de la instrucción de *asignación*

- ❑ La `<variable>` de la parte izquierda de la instrucción debe haber sido declarada previamente.
- ❑ Se evalúa la `<expresión>` de la parte derecha.
- ❑ Se modifica el valor de la `<variable>` con el resultado de la evaluación de la `<expresión>`.

Asignación

```
int m = 3;           // m = 3
int n = m;           // m = 3, n = 3
n = 2 + 7;           // m = 3, n = 9
m = (4 * n) - 2;     // m = 34, n = 9
n = n + 1;           // m = 34, n = 10
```



Otros operadores de asignación

```
n = n + 1;
```

```
n += 1;
```

```
n++;
```

Conversión de tipos

□ Tipos

■ Respecto a la información

- Conversión sin pérdida de información
- Conversión con pérdida de información

■ Respecto a la sintaxis

- Conversión implícita
- Conversión explícita

Ejemplo

```
#include <iostream>
using namespace std;
/*
 * Programa que comprueba qué conversiones
 * automáticas que realiza C++.
 */
int main() {
    int edad;          cout << edad << endl;
    edad = 18;         cout << edad << endl;
    edad = 17.8;       cout << edad << endl;
    edad = "18";       cout << edad << endl;
    edad = true;       cout << edad << endl;
    return 0;
}
```

Ejemplo

```
#include <iostream>
using namespace std;
/*
 * Programa que prueba las conversiones
 * automáticas que
 */
int main() {
    int edad;
    edad = 18;
    edad = 17.8;
    // edad = "18";
    edad = true;
    return 0;
}
```

Advertencia:

Se está usando la variable edad, que no está inicializada

```
cout << edad << endl;
cout << edad << endl;
cout << edad << endl;
cout << edad << endl;
cout << edad << endl;
```

Error:

Conversión no válida de **const char*** (cadena de caracteres) a **int**



Posible resultado de la ejecución

4201088

18

17

1

Otro ejemplo más. ¿Qué está mal?

```
#include <iostream>
/*
 * Programa erróneo que pretende escribir en la
 * pantalla el porcentaje de aprobados
 * correspondiente a 95 estudiantes aprobados con
 * respecto a 160 estudiantes matriculados.
 */
int main() {
    unsigned aprobados = 95;
    unsigned matriculados = 160;

    double porcentaje = aprobados / matriculados * 100;

    std::cout << porcentaje << std::endl;
    return 0;
}
```

¿Cuáles son correctas?

```
unsigned aprobados = 95;  
unsigned matriculados = 160;
```

```
double tasa;
```

```
tasa = aprobados / matriculados;  
tasa = double(aprobados / matriculados);  
tasa = double(aprobados) / matriculados;  
tasa = aprobados / double(matriculados);  
tasa = double(aprobados) / double(matriculados);
```