Grau en Enginyeria en Informàtica

PROGRAMACIÓ I

Tipus de dades simples

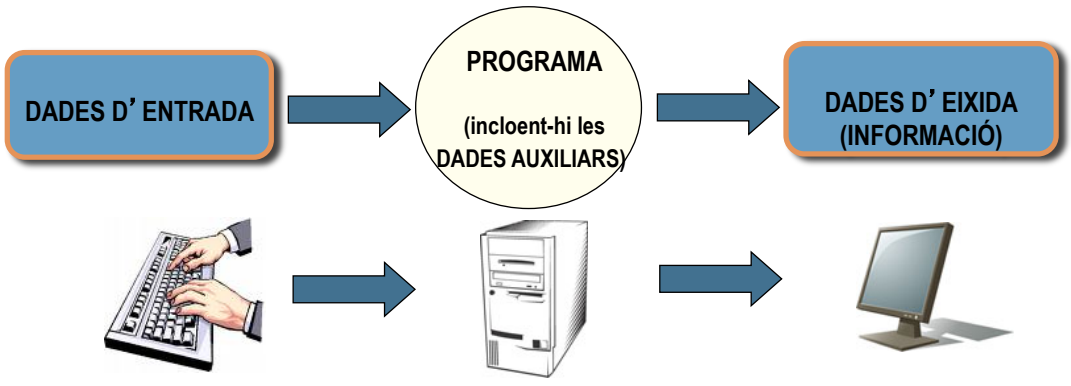
Objectius / competències

1. Comprendre l'ús de dades en un programa.
2. Conèixer els tipus de dades simples d'un llenguatge de programació.
3. Aprendre a usar, llegir i imprimir tipus de dades simples en llenguatge C (C++).

Índex

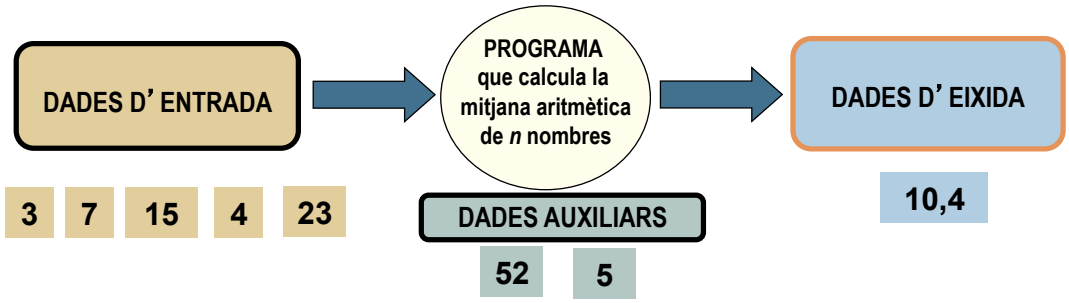
Dades en un programa

- Dada = fet o valor a partir del qual es pot inferir una conclusió (informació)
- Dades en un programa = dades amb què opera una computadora
  - Les dades d'entrada constitueixen un punt de partida per a obtenir coneixement (dades d'eixida).
  - El programa també pot necessitar dades auxiliars (internes) per a obtenir el resultat.



Exemple de dades en un programa

- Programa: calcula la mitjana aritmètica de *n* nombres
- Dades d'entrada: *n* nombres qualsevol
- Dades d'eixida: la mitjana aritmètica dels *n* nombres
- Dades auxiliars :
  - la suma dels nombres
  - la quantitat de nombres

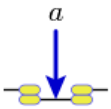


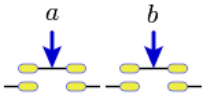
Tipus de dades en un programa

- Tipus de dada = valors + operacions
  - Conjunt de valors que pot prendre una dada en el programa.
    - Si intentem donar-li un valor fora del conjunt pot produir-se un error.
  - Conjunt d'operacions que es poden definir sobre les dades.

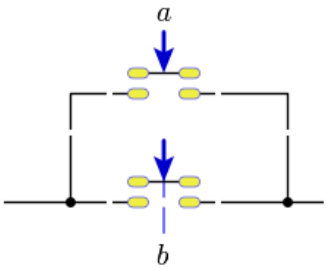
Exemple de tipus de dades

A	B	not A	A and B	A or B
false	false	true	false	false
true	false	false	false	true
false	true		false	true
true	true		true	true





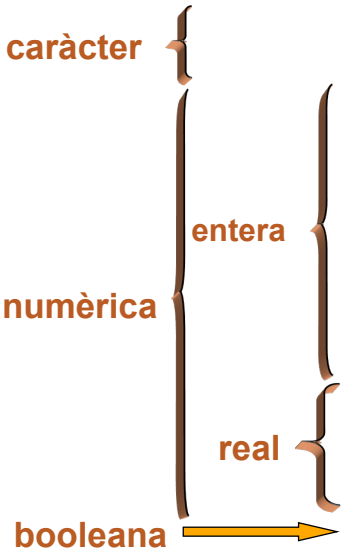
- Tipus de dada booleana
  - Valors = { true, false }
  - Operacions = { and, or, not }



Tipus de dada simple

- Són tipus elementals que deriven d'altres tipus.
- Cada valor concret del tipus de dada simple està especificat per un literal.
  - Per exemple, els literals enters poden expressar-se:
    - En decimal (base 10) : *255*
    - En octal (base 8) : *0377* $(3*8^2 + 7*8^1 + 7 = 255)$
    - En hexadecimal (base 16): *0xff* $(15*16^1 + 15 = 255)$

Tipus de dades simples predefinides en C



Tipus	Significat	Bytes
char	caràcter	1
unsigned char	caràcter sense signe	1
int	enter	2-4
short	enter curt	2
long	enter llarg	4
long long	enter llarg	8
unsigned	enter sense signe	2-4
unsigned short	enter curt sense signe	2
unsigned long	enter llarg sense signe	4
unsigned long long	enter llarg sense signe	8
float	coma flotant (número real)	4
double	coma flotant de doble precisió	8
long double	coma flotant de doble precisió extensa	16
bool	booleana	1

El tipus **bool** no existeix en l'estàndard ANSI C i és emulat amb el tipus **int** (zero és valor *false*, i diferent de zero és valor *true*)

Nosaltres utilitzarem el tipus bool de C++ i de l'estàndard C99

Valors de tipus de dades en C

Tipus	Bytes	Valors	precisió
char	1	Alfabètics: 'a', 'b',…' z' 'A', 'B', …' Z' Dígits: '0', '1', '2', .. '9' Especials: '+', '-', '/', '=', '(', …	
short	2	-32.767..32.767	
int	4	-2.147.483.647..2.147.483.647	
float	4	Aprox. $10^{-38}..10^{38}$	7 dígits
double	8	Aprox. $10^{-308}..10^{308}$	15 dígits
bool	1	true, false	

Tipus de dades enumerades

- Generalment els llenguatges de programació tenen tipus de dades predefinides i, a més, possibiliten a l'usuari definir els seus propis tipus de dades

En el llenguatge C

- L'usuari pot definir tipus de dades enumerades compostes per un conjunt d'identificadors que representen un valor sencer.
- No hi ha format d'impressió per a aquests tipus. El primer element té associat el valor 0, el segon el valor 1 i així successivament.

```
enum T_DiaSemana {dilluns, dimarts, dimecres, dijous, divendres, dissabte, diumenge};

enum T_Color_Primari {roig, verd, blau};
```

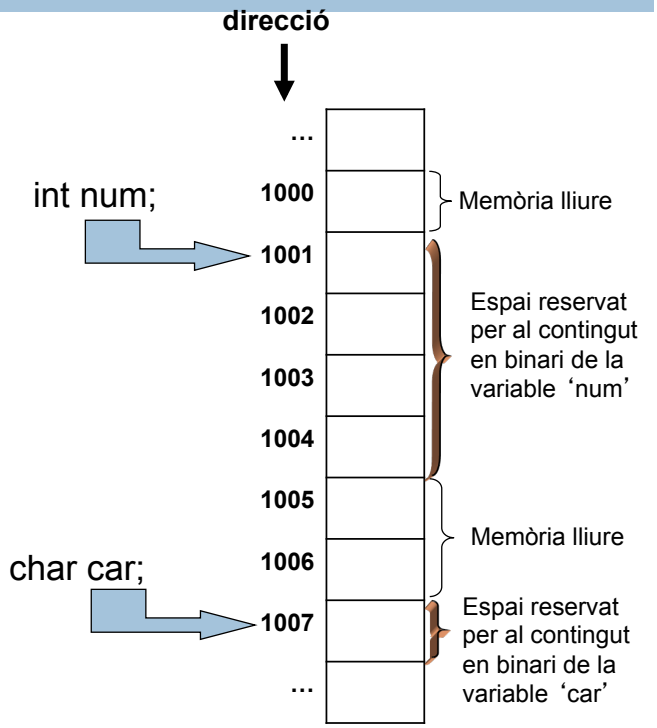
Variables i constants en un programa

- Característiques comunes:
  - Permeten representar dades en un programa.
  - Constitueixen un espai de memòria reservat per a emmagatzemar un valor d'un tipus de dada.
  - S'identifiquen amb un nom.
- Es diferencien en…
  - El valor d'una variable pot canviar al llarg del programa.
  - El valor d'una constant no canvia mai en el programa.

Representació en memòria de les variables

12

- La memòria consisteix en una llista de posicions numerades (bytes).
- Una variable representa una porció de memòria composta per un nombre consecutiu de bytes
- Una variable en memòria està determinada per:
  - La direcció en la memòria que proporciona la ubicació del primer byte dedicat a aquesta variable.
  - El tipus determina quants bytes de memòria requereix la variable.



Exemple de variables i constants

13

Un club de futbol X propietari d'un estadi Y necessita calcular la recaptació de cadascun dels partits disputats en el seu estadi, tenint en compte que posa a la venda tres tipus d'entrada segons el lloc en la grada del seient seleccionat per l'aficionat: entrada de fons (grades darrere de les porteries), entrada general (grades laterals sense sostre) i entrada preferent (grades laterals amb sostre). Durant tota la temporada, el preu d'una entrada de fons és la meitat que el d'una entrada general i el preu d'una entrada preferent és el doble d'una entrada general. En cada partit, el club de futbol fixa un preu per a l'entrada general i, a més, hi ha uns descomptes per a tota la temporada per als xiquets (el 80%) i per als pensionistes (el 50%).

Variable real

Per a calcular la recaptació de cada partit de la temporada…

- Per a emmagatzemar el preu de l'entrada general…
- Per a emmagatzemar el descompte per als xiquets…

Constant

- Per a emmagatzemar el nombre d'entrades preferents venudes…
- i per a emmagatzemar el tipus d'entrada venuda?...
- i per a emmagatzemar si s'aplica descompte o no a un aficionat?...

Variable **entera**

- i per a emmagatzemar el preu d'una entrada preferent?...

Identificadors de variables i constants

- Notacions molt esteses en la majoria de programadors:
  1. Les variables en minúscules i les constants en majúscules

  2. Amb identificadors compostos per diverses paraules:
     - Tot en minúscules i separades les paraules amb el caràcter subratllat
       `nom_alumne`
     - Tot en majúscules i separades les paraules amb el caràcter subratllat
       `NOM_ALUMNE`
     - Tot en minúscules llevat de les inicials de cada paraula
       `NomAlumne`
     - Ús d'abreviatures amb la mateixa longitud
       `nom_alu`

És molt important no canviar arbitràriament de **notació** i seguir-ne només una per a mantenir una coherència en els nostres programes i facilitar la legibilitat i la comprensió.

Identificadors en un programa

- Un identificador és un nom que utilitza el programador per a referenciar les dades i altres elements del programa
- Regles generals de construcció d'identificadors:
  1. Ha de resultar significatiu.
  2. No pot coincidir amb paraules reservades pròpies del llenguatge de programació.
  3. La longitud no ha de ser excessivament llarga.
  4. Han de començar per un caràcter alfabètic o el símbol de subratllat i poden contenir caràcters alfabètics, dígits i el símbol de subratllat.
  5. No s'accentuen.
  6. Segons el llenguatge de programació pot ser utilitzat, indistintament o no, en majúscules o en minúscules.

Els llenguatges C i C++ són **sensibles** a majúscules i minúscules

Exemples d'identificadors

Identificadors **correctes**:

- distancia
- distancia_euclidea
- _data
- dataNaixement
- NUMERO_PI
- numero1
- numero_2

Identificadors **incorrectes**:

- distancia-euclidea
- 3llibres
- Numero$1
- Mes_preguntes?
- número

Aquests **identificadors** són dferents en llenguatge C:

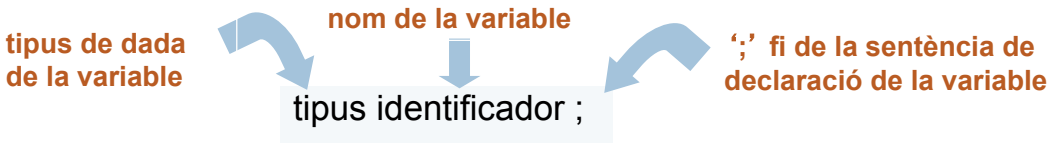
Color_cotxe color_cotxe COLOR_COTXE color_Cotxe Color_Cotxe

Com s'usen les variables i constants?

- Pas 1: cal declarar-les
  - El programador ha d'assignar un nom i determinar el tipus de dada perquè el compilador reserve l'espai de memòria necessari per a emmagatzemar un valor d'aquest tipus de dada.
- Pas 2: cal inicialitzar-les
  - El programador ha d'assignar un primer valor abans que siga usada.
- Pas 3: cal utilitzar-les
  - El programador les ha d'utilitzar en els llocs del programa (sentències) permesos, segons les regles sintàctiques que estableix el llenguatge de programació usat.
- Pas 4: cal destruir-les
  - El compilador allibera l'espai de memòria prèviament reservat
  - Normalment no correspon al programador fer aquest pas, però ha de tenir en compte quan es produeix per a no utilitzar variables i constants una vegada destruïdes.

Declaració de variables en llenguatge C

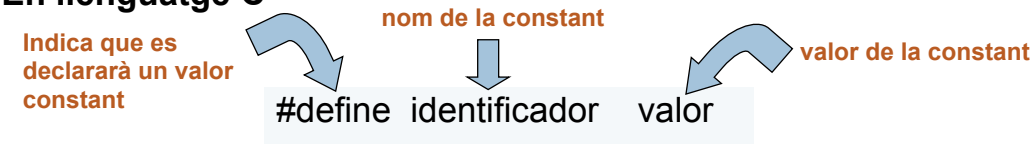
Cal associar un tipus de dada a la variable per tal que s'hi puga emmagatzemar qualsevol valor d'aquest tipus de dada.




```
char  letra_dni;   // variable per a emmagatzemar la lletra del DNI de qualsevol persona
int   paginas;     // variable per a emmagatzemar el nombre de pàgines de qualsevol llibre
float sueldo;      // variable per a emmagatzemar el sou de qualsevol persona
bool  aprobado;    // variable per a emmagatzemar si un alumne ha aprovat o no una assignatura
```

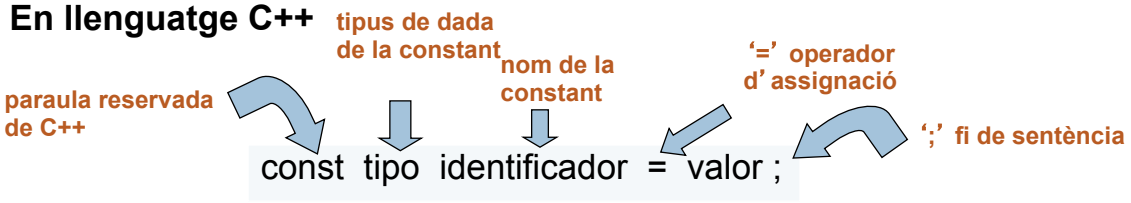
Declaració i inicialització de constants en C / C++

En llenguatge C




```
#define  HORES_DIA  24   // constant que emmagatzema el nombre d'hores d'un dia
```

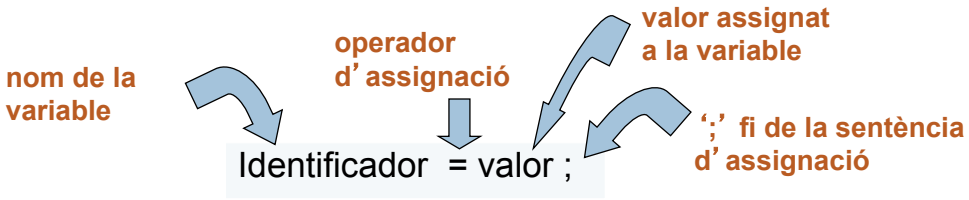
En llenguatge C++




```
const  int  HORES_DIA = 24;   // constant que emmagatzema el nombre d'hores d'un dia
```

Inicialització de variables en llenguatge C

Mitjançant la **sentència d'assignació** :



En llenguatge C

```
letra_dni = 'A' ;   // emmagatzema el caràcter 'A' en la variable lletra_dni
pagines = 365;  // emmagatzema la quantitat 365 en la variable pàgines
sou = 1000.20;     // emmagatzema la quantitat real 1000.20 en la variable sou
aprovat = true;   // emmagatzema el valor true en la variable aprovat
```

En C i C++ es permet **inicialitzar** variables en la declaració, per exemple: int pagines = 365;

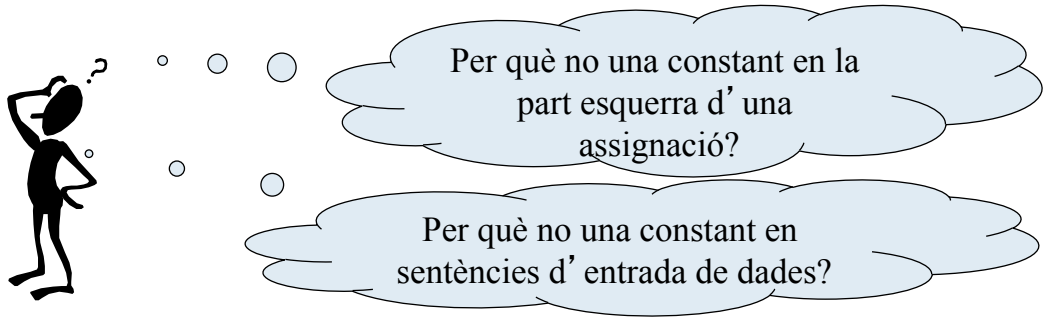
Utilització de variables i constants en llenguatge C

Una variable s'utilitza…

- en la part esquerra d'una sentència d'assignació
- en una expressió aritmètica o lògica
- en sentències d'entrada i eixida de dades

Una constant s'utilitza…

- en una expressió aritmètica o lògica
- en sentències d'eixida de dades



Sintaxi de la sentència d'assignació

En llenguatge C

```
pagines_llibreA = 430; // emmagatzema el número 430 en la variable pagines_llibreA, declarada prèviament de tipus int
sou = 35616.44;    // emmagatzema el número real 35616.44 en la variable sou, declarada prèviament de tipus float
```

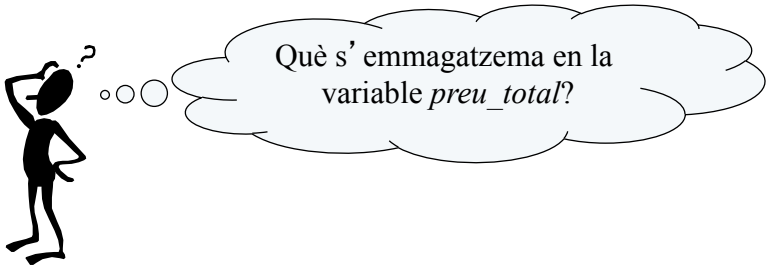
Com funciona la sentència d'assignació?

Pas 1: s'avalua la part dreta de l'operador d'assignació

Pas 2: s'assigna el valor de la part dreta a la variable de la part esquerra de l'operador d'assignació

En llenguatge C

```
// si suposem que s'han declarat prèviament les variables relacionades amb preus de tipus float
preu_cotxeA = 10500.00;  // emmagatzem la quantitat 10500.00 en la variable preu_cotxeA
preu_cotxeB = 40200.00;  // emmagatzem la quantitat 40200.00 en la variable preu_cotxeB
preu_total = preu_cotxeA + preu_cotxeB;
```



Expressions aritmètiques i lògiques

Una expressió en un programa és una combinació de variables, constants, operadors, parèntesis i identificadors de funcions; avaluant-los s'obté un valor.

- Les expressions es poden escriure en qualsevol lloc del programa en què puga utilitzar-se el valor que tornen

Una expressió aritmètica…

- es construeix amb operadors aritmètics
- torna un valor numèric

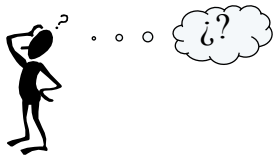
```
(x_rad * 360) / (2 * PI)
```

calcula els graus corresponents al valor en radians emmagatzemats en la variable x_rad, utilitzant la constant PI

Una expressió lògica…

- es construeix amb operadors relacionals i lògics
- poden aparèixer operadors aritmètics
- torna un valor booleà

```
(any mòdul 4 == 0) AND (NOT (any mòdul 100 == 0) OR (any modul 400 == 0) )
```



Operadors en llenguatge C

operadors aritmètics	significat	tipus d'operands	tipus de resultat
+ - * /	suma, resta, multiplicació, divisió	numèrics **enters** o reals	numèric **enters** o real
%	resta de divisió	**enters**	**enters**
operadors relacionals			
< > <= >=	menor que, major que, menor o igual que, major o igual que	tipus simples	Booleà
== !=	igual que, diferent de	tipus simples	booleà
operadors lògics			
&&	AND lògic	booleà	booleà
||	OR lògic	booleà	booleà
!	NOT lògic	booleà	booleà

Cal tenir clara la diferència entre l'**operador d'assignació** '=' i l'operador relacional d'**igualtat** '==‘. És habitual utilitzar erròniament l'operador '=' en lloc de '==', cosa que provoca errors difícils de detectar

Amb l'**operador de divisió** '/', quan els operands són de tipus numèric sencer, el resultat és la part sencera del quocient. Per a obtenir un resultat amb decimals, algun dels operands ha de ser de tipus numèric real.

Precedència i associativitat d'operadors

operadors	significat	associativitat
- !	signe negatiu d' un número, NOT lògic	de dreta a esquerra
* / %	multiplicació, divisió, resta	d' esquerra a dreta
+ -	suma, resta	d' esquerra a dreta
< > <= >=	de relació	d' esquerra a dreta
== !=	d' igualtat	d' esquerra a dreta
&&	AND lògic	d' esquerra a dreta
||	OR lògic	d' esquerra a dreta

Ordre de precedència dels operadors en llenguatge C

La **precedència** o **prioritat** d'un operador indica l'ordre en què s'executen les operacions en una expressió que conté diferents operadors

L'**associativitat** d'un operador indica l'ordre en què s'executen les operacions en una expressió que conté operadors amb la mateixa prioritat

És recomanable l'ús de **parèntesis**:
- quan tenim algun dubte de l'ordre d'avaluació
- per a fer-la més llegible
- per a modificar l'ordre d'avaluació

Sentències d'entrada i eixida de dades

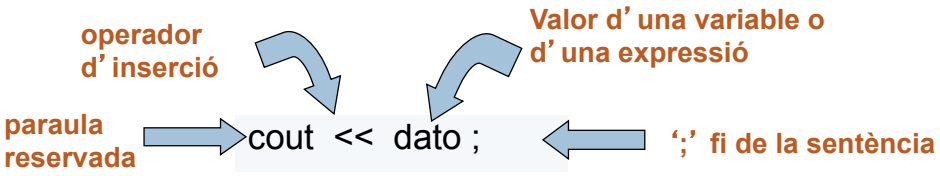
- Les variables també poden utilitzar-se en sentències d'entrada.
- Les variables, constants i en general les expressions també poden utilitzar-se en sentències d'eixida.
- Les **sentències d'entrada** permeten emmagatzemar en variables dades que l'usuari introdueix per teclat.
- Les **sentències d'eixida** permeten visualitzar dades en la pantalla

L'entrada i eixida poden estar associades a diferents **fonts i dispositius** com ara fitxers, impressores, pantalles tàctils, ratolí, etc.

En aquesta assignatura, només usarem en els programes el **teclat i la pantalla** que solen ser els dispositius d'entrada i eixida per defecte.

Sentència d'eixida en C++

- Permet escriure en pantalla qualsevol combinació de valors de variables, constants, expressions i cadenes de text



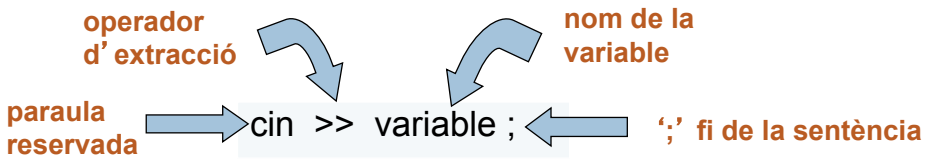
Exemples

```
cout  << "El preu de l'ordinador portàtil és de " << preu << " euros" << endl;
cout << "el preu total és : " << (preu1 + preu2);
cout << preu;
cout << "això és una cadena de text sense salt a una nova línia";
cout << "això és una cadena de text amb salt a una nova línia\n";
cout << endl;
cout << "\n";
```

En llenguatge C per a l'eixida de dades s'utilitza la funció de llibreria **`printf()`**, però optem per utilitzar la sentència **`cout`** de C++ perquè és més senzilla d'usar

Sentència d'entrada en C++

- Permet emmagatzemar en variables valors introduïts per teclat



Exemples

```
cout  << "Introdueix la teua edat:";
cin >> edat;   // edat serà una variable declarada de tipus int
cout << "introdueix les notes dels dos parcials de pràctiques:";
cin >> nota1 >> nota2;  // nota1 i nota2 seran variables declarades de tipus float o double
cout  << "Vols introduir més dades? (S/N) : ";
cin >> resposta;   // resposta serà una variable declarada de tipus char
```

En llenguatge C per a l'entrada de dades s'utilitza la funció de llibreria **`scanf()`**, però optem per utilitzar la sentència `cin` de C++ perquè és més senzilla d'usar

`cin` ignora espais en blanc i el caràcter fi de línea

Quina classe de programes he de ser capaç de fer?

#directives del preprocessador

Declaració de constants

main() {

Declaració de variables:

de tipus simples

Cos principal (sentències executables)

sentències d'entrada i eixida

sentències d'assignació

}

```
#include <iostream>
using namespace std;
const float kPI =  3.1415926;

main() {

  float area, radi;

  cout << "Introdueix el radi del cercle:";
  cin >> radi;
  area = kPI * radi * radi;
  cout << "L'àrea del cercle és:" << area;
  cout << endl;
}
```

Declaració i inicialització d'una constant

Declaració de dues variables

Totes les sentències en C i C++ acaben amb un **punt i coma**

Bibliografia recomanada

Fundamentos de Programación
Jesús Carretero, Félix García i altres
Thomson-Paraninfo 2007. ISBN: 978-84-9732-550-9

✓ Capítol 2 (apartat 2.4)
✓ Capítol 4 (apartats 4.1; 4.2; 4.3; 4.4; 4.10)

Problemas Resueltos de Programación en Lenguaje C
Félix García, Alejandro Calderón i altres
Thomson (2002) ISBN: 84-9732-102-2

✓ Capítol 2 (apartats 2.1; 2.2; 2.3)

Resolución de Problemas con C++
Walter Savitch
Pearson Addison Wesley 2007. ISBN: 978-970-26-0806-6

✓ Capítol 2