3. Variables (i constants) i la seva definició

Introducció

1. Sintaxi d'un programa

2. Definició de variables

3. Assignació de variables

4. Definició de constants

5. Abast i variables globals

Resum

Introducció

Ja heu conegut els tipus de dades elementals que es poden utilitzar en un programa. No obstant això, si us hi fixeu, de moment només sabem fer operacions entre dades d'un tipus, com si es tractés d'una calculadora, o comparar diferents valors entre si. Però això no és el que es busca; el que cal és definir maneres de tractar dades, i en el cas que canviïn, processar-les de la mateixa manera sense haver de fer res més.

Per a això hi ha les variables. En aquest apartat, aprendreu el que són les variables i com es defineixen primer en pseudocodi i, després, en un programa en C.

Tornem al punt en què s'introduïen les taules de les funcions lògiques (NOT, AND i OR). S'hi parlava de **a** i **b**, que representaven dos valors booleans que podien ser `true` o `false`. Fixeu-vos, però, que a i b no tenen un valor específic (tots dos poden ser `true` o `false`). Per tant, a i b són variables.

Una **variable** és un nom simbòlic que fa referència a una informació concreta o a unes dades que es guarden a la memòria. Quan es treballa amb variables, en comptes de fer-ho amb un valor concret, s'escriuen instruccions que haurien de funcionar per a qualsevol valor que prengui aquesta variable.

Ara bé, una variable no es pot considerar únicament com una adreça de memòria, ja que té més informació associada. En parlar de la memòria, hem explicat breument que la informació es guarda en un dispositiu utilitzant un format de 0 i 1. Però guardar la informació en un ordinador és una mica més complicat que guardar 0 i 1. És a dir, com se sap que un 0 o un 1 pertany a un nombre de 8 bits o a un de 64 bits de mida? Com s'estructura la memòria per tenir en compte aspectes com aquest?

Per donar resposta a aquestes preguntes, es defineixen les variables en els programes. Quan s'executa un programa, aquest demana un espai de memòria a l'ordinador i el gestiona guardant variables de diferents mides en aquest espai. En tot moment, el programa ha de saber quina variable es guarda a cada part de la memòria que el processador li ha assignat. És a dir, a cada variable se li assigna un espai de la memòria que té una adreça, de manera que el programa sap en tot moment el que té guardat i a on.

Una variable té, com a mínim, els atributs o les característiques següents:

- **Nom o identificador.** Una variable pot tenir qualsevol nom simbòlic que se li vulgui donar. És recomanable que el nom d'una variable ajudi a entendre el tipus d'informació que guarda. Per exemple, si volem guardar el saldo d'un compte bancari, podríem anomenar la variable *saldo*; i si volem guardar el cost d'un bé, podríem anomenar-la *cost*. Cal tenir en compte que, en els dos exemples, el valor que guardarà la variable serà un nombre real, però el nom ajudarà a entendre el significat del contingut de la variable dins del programa.
- **Tipus de dada que la codifica**. Com ja sabeu, pot ser boolean (`bool`), integer (`int`), real (`float`) o char (`char`).
- **Valor.** El valor guardat d'una variable és el valor concret que té en un moment determinat de l'execució del programa. El valor canviarà segons les operacions que s'hi facin, i sempre haurà de ser un valor dins del rang de valors vàlids que pot prendre.
- **Adreça de memòria.** L'adreça de memòria és la posició en què el valor de la variable està guardat. Internament, durant l'execució del programa, l'adreça s'utilitza per saber on guardar i trobar el valor de la variable en cada moment.
- **Visibilitat.** Més endavant, veurem que un algorisme està format per diferents parts. Les variables només seran visibles en determinades parts de l'algorisme, en funció d'on estiguin definides.

Quan s'escriu un programa, cal descriure cadascuna de les variables que s'utilitzen; és a dir, si s'usarà una variable que s'anomena **a** i que és de tipus booleà, cal explicitar tant el nom com el tipus de dada que la variable guardarà. Això es fa mitjançant la **definició** de la variable.

És important recordar que, quan creem una variable, se li pot assignar un valor que pot dependre dels valors que hi hagi a la memòria en el moment de la creació. Per exemple, si quan es defineix una variable se li assigna un espai de memòria en el qual ja hi havia valors de 0 i 1, pot ser que la variable es generi amb un valor inicial que no coneguem. Per això, és important que, sempre que es pugui, s'**inicialitzi la variable en crear-la**, és a dir, que se li assigni un valor que tingui sentit.

En aquesta unitat, explicarem com es defineixen i s'inicialitzen les variables.

1. Sintaxi d'un programa

Quan s'escriu un programa, cal fer-ho de manera que el compilador ho entengui, és a dir, cal seguir una **sintaxi** molt estricta. Si no es fa així, el compilador no entendrà el codi que s'ha escrit, i produirà un error. Aprendre a programar consisteix a dominar una sèrie de conceptes (com el de les variables que acabem d'explicar), però també en saber implementar-los o escriure'ls en un programa perquè es puguin transformar en un programa executable. A continuació, aprendreu la primera sintaxi d'un programa C.

2. Definició de variables

Per definir una variable d'un tipus de dada, cal seguir la sintaxi que s'introdueix a continuació. La definició depèn del tipus de dades. Seguidament, hi incloem totes les opcions:

Pseudocodi	C

var
booleanVariable: **boolean**;
integerVariable: **integer**;
realVariable: **real**;
charVariable: **char**;
end var
	`#include <stdbool.h>`
	
	`int main() {`
	`bool booleanVariable;`
	`int integerVariable;`
	`float realVariable;`
	`char charVariable;`
	`}`

Fixeu-vos en un parell de coses:

- El tipus de variable `bool` en C no és un tipus natiu, és a dir, no està definit en C tal qual. Per poder utilitzar les variables `bool` en C, tal com us expliquem en aquest curs, heu d'escriure la línia `#include <stdbool.h>`. Amb aquesta línia afegim noves funcionalitats a C. En aquest cas, hi afegim la possibilitat d'usar variables booleanes i els podem assignar els valors `true` o `false`.
- Els noms de les variables han de ser únics: no es pot anomenar dues variables amb el mateix nom, encara que siguin de tipus diferents. Per això, els noms que hem triat comencen amb el tipus de variable. Tanmateix, en crear una variable, el nom pot ser el que vulgueu.

Recordeu

Per usar una variable de tipus `bool` en C, heu d'afegir la línia `#include <stdbool.h>` al principi del codi.

Es poden definir diverses variables del mateix tipus de la manera següent:

Pseudocodi	C

var
booleanVariable1, booleanVariable2: **boolean**;
integerVariable1, integerVariable2: **integer**;
realVariable1, realVariable2: **real**;
charVariable1, charVariable2: **char**;
end var
	`#include <stdbool.h>`
	
	`int main() {`
	`bool booleanVariable1, booleanVariable2;`
	`int integerVariable1, integerVariable2;`
	`float realVariable1, realVariable2;`
	`char charVariable1, charVariable2;`
	`}`

Hi ha una sèrie de normes que s'han de seguir en triar un nom per a una variable:

- Els noms poden contenir lletres, dígits i guions baixos, però no espais en blanc ni caràcters especials com !, #, ç, ñ, %, etc.
- Els noms han de començar amb una lletra o un guió baix (_)
- Els noms distingeixen entre majúscules i minúscules (myVar i myvar són variables diferents)
- Les paraules reservades (com ara int) no es poden utilitzar com a noms.

Per convenció, els noms de les variables s'escriuen en minúscula. Si el nom de la variable està format per dues paraules, la primera lletra del segon terme s'escriurà en majúscula. Això és el que s'anomena **camelCase (https://en.wikipedia.org/wiki/Camel_case)** .

En el pseudocodi, apareixen una sèrie de paraules en negreta. Aquestes paraules formen part de la sintaxi del pseudocodi. En el codi, en canvi, apareixen acolorides. Aquestes paraules serveixen perquè el compilador (l'intèrpret del codi escrit) entengui a cada moment què es vol fer. Es diuen **paraules clau** (o reservades) del llenguatge i no es poden utilitzar com a nom ni com variables o constants. Cada llenguatge de programació té les seves paraules clau, encara que la majoria coincideixen entre els diferents llenguatges.

D'ara endavant, sempre que es vegin paraules en negreta en un algorisme significarà que són paraules clau del llenguatge algorítmic i es podran distingir de la resta.

3. Assignació de variables

Anteriorment, hem dit que és aconsellable que, un cop definida una variable, s'inicialitzi. Per fer-ho, cal assignar-li un valor. Assignem un valor a una variable de la manera següent:

Pseudocodi	C

var
booleanVariable: **boolean**;
integerVariable: **integer**;
realVariable: **real**;
charVariable: **char**;
end var
 	
booleanVariable := **false**;
integerVariable := 0;
realVariable := 0.0 ;
charVariable := 'a';
	`#include <stdbool.h>`
	
	`int main() {`
	`  bool booleanVariable;`
	`  int integerVariable;`
	`  float realVariable;`
	`  char charVariable;`
	
	`  booleanVariable = false;`
	`  integerVariable = 0;`
	`  realVariable = 0.0;`
	`  charVariable = 'a';`
	`}`

En pseudocodi, per assignar un valor a una variable s'utilitza l'**operador d'assignació** :=, que es llegeix «presa per valor».

En C, s'utilitza l'operador = .

En aquest punt, us volem fer uns quants aclariments pel que fa a l'assignació de valors a variables en C. En concret, amb variables de tipus `bool`. Fixeu-vos en l'exemple anterior: hem definit la variable `booleanVariable`; després, li hem assignat el valor `true`, i, en la línia següent, li hem assignat el valor `false`:

```
#include <stdbool.h>

int main() {
    bool booleanVariable;

    booleanVariable = true;
    booleanVariable = false;
}
```

Compte!

Les paraules `true` i `false` són reservades en C i són els valors que pot prendre una variable `bool`. En introduir les variables booleanes en **2. Tipus elementals de dades (https://aula.uoc.edu/courses/78741/pages/2-tipus-elementals-de-dades-2-2)**, vam explicar que `false` es

pot relacionar amb 0 i `true` amb 1. Però no ens hem d'equivocar en escriure el codi en C: no hem d'assignar els valors 0 o 1 a una variable `bool`, sinó que hem d'assignar només els valors `true` o `false`. Tampoc no cal emprar cometes abans i després de `true` o `false`. Això també seria un error, que entendrem millor quan estudiem les cadenes de caràcters.

4. Definició de constants

De vegades, en un programa, s'utilitzen valors o paràmetres que són rellevants perquè funcioni. Aquests paràmetres poden ser nombres i, en general, pot ser que no sigui fàcil identificar-los d'un cop d'ull. Per exemple, és més fàcil identificar el número 42 si diem que és el nombre de plantes d'un hotel (NUM_FLOORS) o 3.14159 si diem que és el nombre π (pi).

Si per emprar aquests paràmetres en un programa utilitzem una constant en comptes de posar el nombre en cada fórmula, aconseguirem dues coses:

- El nom de la constant ens diu quin nombre estem codificant i és més fàcil d'identificar en operacions o fórmules.
- Si cal canviar el nombre de plantes d'un hotel de 42 a 43, ho farem en la definició de la constant, i quedarà ja canviat totes les vegades que s'usarà en el programa (en comptes d'anar a cada línia en què consta 42 i canviar-ho per 43).

La definició d'aquestes constants es fa de la manera següent:

Pseudocodi	C

const
PI: **real** = 3.1415926;
MAX_STUDENTS: **integer** = 50;
IVA: **integer** = 21;
BLANK_CHAR: **char** = ' ';
end const
	`#define PI 3.1415926`
	`#define MAX_STUDENTS 50`
	`#define IVA 21`
	`#define BLANK_CHAR ' '`

5. Abast i variables globals

Potser no us hàgiu adonat d'un detall molt important: totes les variables que hem definit en els programes C presentats en aquesta unitat estaven dins del programa principal `main`. Això té una raó que cal comentar, tot i que ara sigui difícil d'entendre.

En C, quan una variable es defineix entre claus, {}, el programa considera que aquesta variable només es pot utilitzar entre aquestes claus. Així doncs, quan definim una variable en el programa `main`, és a dir, entre les claus que defineixen aquest programa, la variable no es pot utilitzar fora d'aquestes claus. Això és el que s'anomena **abast** d'una variable (en anglès, *scope*); és a dir, les línies de codi dins d'una d'aquestes claus en les quals es pot utilitzar la variable que s'ha definit.

L'**abast** (*scope*) d'una variable és el context en el qual la variable és definida, és a dir, les línies de codi en les quals la definició de la variable és correcta o vàlida. En C, l'abast d'una variable ve determinada per les claus, {}, en les quals està definida.

Us demanem que no ho feu, però heu de saber que és possible definir variables fora de la funció `main`. Si us hi fixeu, fora de la funció `main` no hi ha claus, {}, per tant, si definim una variable fora d'aquesta funció, el seu abast serà tot el programa. Aquestes són les anomenades **variables globals**.

Les **variables globals** són aquelles que tenen un abast que engloba tot el programa. Estan definides fora de la funció `main` (o de qualsevol altra funció).

Cal evitar les variables globals.

Com us dèiem, tot i que ara no és fàcil de veure la raó, l'ús de variables globals pot portar problemes: pot ser que desenvolupem una part del codi i definim una variable que pensem que no existeix, però que algú altre hagi creat una variable global amb aquest nom. No sembla un problema greu, però la veritat és que pot tenir conseqüències poc desitjables, així que insistim que no ho feu i que definiu les variables només dins de la funció `main` (o una altra funció que genereu, però això ja ho veurem en el futur).

Resum

- Una **variable** és un nom simbòlic que fa referència a una informació concreta o a unes dades que es guarden a la memòria. Quan treballem amb variables, en comptes de fer-ho amb un valor concret, s'escriuen instruccions que haurien de funcionar per a qualsevol valor que prengui aquesta variable.
- Una variable consta dels atributs o les característiques següents:
  - **Nom o identificador.**
  - **Tipus de dada que la codifica**.
  - **Valor.**
  - **Adreça de memòria.**
  - **Visibilitat.**
- Per utilitzar una variable, cal **definir-la** seguint la sintaxi adequada i **inicialitzar-la** amb un valor.
- Per inicialitzar una variable o per canviar-ne el valor en tot un programa, cal poder **assignar-li** valors, sigui directament o mitjançant una operació.
- De vegades, caldrà definir constants, és a dir, dades que no canvien al llarg del programa. És important recordar que és una bona pràctica definir constants quan s'utilitzen nombres (com ara pi o altres paràmetres) en les operacions d'un programa. Així, serà més fàcil canviar-los en tot el programa si fos necessari.
- L'**abast** (*scope*) d'una variable és el context en el qual la variable és definida.
- Cal evitar les **variables globals**, que són aquelles que tenen un abast que engloba tot el programa.