

FONAMENTS D'ORDINADORS

TEMA1: Començant a programar

Manel Guerrero

Algorismes i programes

- Algorisme: seqüència d'instruccions, no ambigües, l'execució de les quals condueix a una resolució d'un problema.
- L'algorisme pot rebre una “entrada”
- I donar com a resultat una “sortida” (la solució del problema).
- Programa: Algorisme codificat (escrit) en un llenguatge de programació determinat.

Estructura d'un computador

- El processador central executa programes
 - que llegeixen de dispositius d'entrada
 - (com ara el teclat)
 - i escriuen a dispositius de sortida
 - (com ara la pantalla)
 - pot emmagatzemar dades en memòria
 - volàtil (memòria RAM)
 - Permanent (disc dur)

Llenguatges de programació

- D'alt nivell (propers a com pensem els humans). Com ara el “C”. Per qualsevol ordinador. [`a = a + 2;`]
- De baix nivell (propers a com funcionen un tipus de computadora en concret) com ara el llenguatge assemblador del I32. [`ADDL $2, %EAX`]
- Codi màquina: Seqüència de '0's i '1's en que un programa es codifica perquè un ordinador en concret ho pugui executar. [`Hex: 83 C0 02`]

Elaboració d'un programa: Etapes

- Anàlisi: Definir bé el problema (ex: programa que donada una base i una alçada calculi l'àrea d'un triangle rectangle).
- Disseny: Desenvolupar l'algorisme
- Codificació: Escriure el codi que l'implementi
- Proba: Execució del codi amb diferents entrades.

- Manteniment: Millora del codi, esmena de nous errors que es van detectant.

Variables

- Objecte identificat amb un nom, que és d'un tipus determinat (ex: enter) i que guarda un valor que es pot anar variant i consultant.
- tipus_var nom_var = valor_inicial;
 - int i; /* declaració */
 - int i, j, k; int a=2, b=3;
 - i = 0; /* assignació d'un valor */
 - int i = 0; /* inicialització (declaració + assignació d'un valor inicial) */

Tipus de variables

- Caracter:
 - char c = 'A';
- Entera:
 - int i = 7;
- Coma flotant:
 - float f = 7.5;
- Coma flotant d'alta precisió:
 - double pi = 3.141592653589793;

Nom d'una variable

- Compost per lletres, números i “_”.
- No pot començar per número.
- “Case sensitive” (diferencia entre majúscules i minúscules). Var, VAR, var són noms de variables diferents.
- Típicament tot en minúscules i usant el “_” enlloc de l'espai. Ex: “nom_variable”.

Constants

- Identificador que representa un valor constant. Es pot consultar però no modificar.
- `#define nom_constant valor_constant`
 - `#define TRUE 1 /* Enter */`
 - `#define FALSE 0`
 - `#define PI 3.1415 /* Decimal */`
 - `#define YES 'Y' /* Caràcter */`
- Les línies de codi que comencen amb '#' no acaben en ';'.
- Els noms de les constants segueixen les mateixes normes que els de les variables.
- Però, típicament tot en majúscules i usant el "_" enlloc de l'espai.

Expressions

- Són formules formades per variables, constants, operadors i parèntesis que es poden avaluar donant com a resultat un valor.
- Operadors:
 - Aritmètics:
 - Suma [+] i Resta [-]
 - Multiplicació [*] i Divisió [/]
 - Residu [%]
 - Increment [++]
 - Decrement [--]
 - Relacionals:
 - Més gran que [>]
 - Més petit que [<]
 - Igual que [==]
 - Diferent que [!=]
 - Més gran o igual [>=]
 - Més petit o igual [<=]
 - Lògics:
 - AND [&&]
 - OR [||]
 - NOT [!]

Expressions: Exemples

- Fals = 0. Cert != 0
- Longitud cercle:

```
#define PI 3.1415
float radi=3, lc;
lc = PI * 2 * radi;
```
- El punt (x,y) pertany a cercle de radi 'r'?

```
int p;
float x=1, y=-2, r=2.23;
p = (x*x + y*y == r*r);
```
- Àrea triangle:

```
float b=2, a=3, at;
at = b*a/2;
```
- És 'n' parell?

```
int n = 3, p;
p = !(n%2);
```
- Les bruixes es pentinen?

```
#define TRUE 1
int plou=TRUE, sol=TRUE;
int lbep;
lbep = plou && sol;
```

Prioritat d'operadors

Operator Type	Operator	Associativity
Primary Expression Operators	() [] . -> expr++ expr--	left-to-right
Unary Operators	* & + - ! ++expr --expr (typecast)	right-to-left
Binary Operators	* / % + - < > <= >= == != && 	left-to-right
Assignment Operators	= += -= *= /= %=	right-to-left

From: www.swansontec.com/sopc.html

- Operators higher in the chart have a higher precedence.
- The "Associativity" column on the right gives their evaluation order.

[H2] Exemples d'expressions

- `a * x * x + b * x + c /* paràbola */`
- `(b1 + b2) * h / 2 /* àrea trapezoide */`
- `!error && calcul_finalitzat`
- `int a=2, b=3;`
- `a++ + b++` retorna 5 (Trampa mortal!)
- `a <= b` retorna 1 (cert).
- `a == b` retorna 0 (fals).
- `(1 && 0) || (1 || 0)` retorna 1 (cert)

Exemples d'expressions 2

- `int a=2, b=3; /* Enters */`
- `b / a` retorna 1 (Trampa mortal!)
- `float a=2, b=3; /* Coma flotant */`
- `b / a` retorna 1.500000
- `float a=2; int b=3; /* var!=0 és cert */`
- `b && a` retorna 1 (Perill!)

Sentencies

- Una sentència ordena una acció.
- Les sentències d'assignació assignen un valor a una variable.
 - `nom_var = expressió;`
 - `i = 0;`
 - `y = a * x * x + b * x + c;`
- Totes les sentències acaben en “;”

Crides a funció

- Ara no és massa important. Però, un altre tipus de sentències són les crides a funció.
- Una funció és una seqüència de sentències identificades mitjançant un nom que realitzen una tasca determinada.
 - `var_retorn = nom_funcio(parametre1, ...);`
 - `int max, a, b;`
 - `max = maxim(a,b);`

printf

- Per imprimir missatges per pantalla

```
#include <stdio.h>

int res;

res = 2 + 3;

printf("El resultat és %d\n", res);

• "\n" salt de línia | "\t" tabulat | "%%" %

printf("El %d %% de %d és %d\n",
      percent, num, res);
```

printf: nivell pro

- %+d, %+f // mostra el signe sempre
- int a=64; printf("a=%6d \n",a);
// escriu "a= 64"
- int hh=0, mm=3, ss=6; printf("%02d:
%02d:02d\n", hh,mm,ss);
// escriu "00:03:06"
- %.2f : imprimeix el número amb dos decimals
- %8.2f : longitud total de 8 dígit, amb dos decimals

scanf

- Per llegir dades per teclat

```
#include <stdio.h>

int dada;

scanf("%d", &dada);

• Llegeix un enter del teclat i l'assigna a la variable dada.

• Si introduïm caràcters no numèrics retorna un -1, no s'assigna res a  
dada i els caràcters introduïts es llegiran de nou en el proper scanf.  
Trampa mortal!

int i1, i2, i3;

scanf("%d %d %d", &i1, &i2, &i3);

• Llegeix tres enters separats per espai i els assigna a i1, i2 i i3.
```

Tipus de dades

Declaració	Rang	printf/scanf
unsigned char	[0, 255] (1 Byte)	%hhu
unsigned short	[0, 65.535] (2 Bytes)	%hu
unsigned int	[0, 2³² -1] (4 Bytes)	%u
unsigned long	[0, 2 ³² -1] (4 Bytes)	%lu
short	[-32.768, 32.767] (2 Bytes)	%hd
int	[-2³¹, 2³¹ -1] (4 Bytes)	%d
long	[-2 ³¹ , 2 ³¹ -1] (4 Bytes)	%d
char	Taula ASCII (1Byte)	%c
float	[1.7E-38, 3.4E38] (4 Bytes)	%f %e
double	[2.22E-308, 1.79E308] (8 Bytes)	%lf %le
long double	(16 Bytes)	%Lf %Le

scanf: de diferents tipus

```
#include <stdio.h>
int i; char c; float f; double d;
unsigned int ui; unsigned char uc;
scanf("%d", &i); scanf("\n%c", &c);
scanf("%f", &f); scanf("%lf", &d);
scanf("%u", &ui); scanf("%hhu", &uc);
scanf("%*c"); // llegeix char i "el llença"
```

- El "\n" en el cas del char és per que si hi han espais en blanc o retorns de carro abans del caràcter se'ls salti (en el cas de "%d" i "%f" això es fa automàticament).

Taula ASCII

000	(nul)	016	▶ (dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	⊙ (soh)	017	◀ (dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	⊕ (stx)	018	‡ (dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	♥ (etx)	019	‡‡ (dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	♦ (eot)	020	‡‡‡ (dc4)	036	\$	052	4	068	D	084	T	100	d	116	t
005	♣ (enq)	021	§ (nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	♠ (ack)	022	— (syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	• (bel)	023	‡ (etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	▣ (bs)	024	↑ (can)	040	(056	8	072	H	088	X	104	h	120	x
009	(tab)	025	↓ (em)	041)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	— (eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	♁ (vt)	027	← (esc)	043	+	059	;	075	K	091	[107	k	123	{
012	⌘ (np)	028	L (fs)	044	,	060	<	076	L	092	\	108	l	124	
013	(cr)	029	↔ (gs)	045	-	061	=	077	M	093]	109	m	125	}
014	♯ (so)	030	▲ (rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	✱ (si)	031	▼ (us)	047	/	063	?	079	O	095	_	111	o	127	o

Espai 32 | 0 48 | A 65 | a 97

Conversions de tipus

- Implícita
 - C permet assignar expressions d'un tipus a variables d'un altre tipus.
 - char <- int: `c = 48; /* c = 48 % 256*/`
 - int <- char: `i = 'a'; /* i = 97 */`
 - int <- float: `i = 3.5; /* i = 3 (trunca) */`
 - float <- double (arrodoneix)
- Explícita
 - `c = (char) 48;` c és igual a '0'
 - `int a=2, b=3; float f = (float)b / (float) a;` f és igual a 1.500000

[H3] Estructura d'un programa en C

- /* llibreries */ (#include)
- /* definició de constants */ (#define)
- /* definició de tipus de dates */ (globals)
- /* prototipus de funciones */ (???)
- /* main : funció principal */ (principi)
- /* definició funcions*/ (???)

Exemple 1

```
#include <stdio.h>

#define PORCENTAJE 20
/* calcula el 20% de un valor */
main() {
    int dato, res;

    printf("Introduce un valor: ");
    scanf("%d", &dato);
    res = dato * PORCENTAJE/100;
    printf("El %d %% de %d es %d \n", PORCENTAJE, dato, res);
}
```

Exemple 2

```
#include <stdio.h>

/* Pasar de minúsculas a mayúsculas: c = c -('a'-'A'); */
main() {
    char c;

    printf("Introduce un carácter: ");
    scanf("%c", &c);
    c = c + 'a' - 'A';
    /* c = (char) ( (int)c - (int)'a' + (int)'A' ) */
    printf("El carácter en minúscula es %c\n", c);
}
```

Exercicis

- A classe. Fer un programa que:
 - Càlculi l'àrea d'un triangle rectangle (amb printf i scanf).
 - Passi un caràcter de majúscules a minúscules.
 - Digui si un caràcter és una lletra minúscula.
 - O una lletra majúscula. O un dígit.
 - Intercanviï el valor de 2 variables.
 - Roti el valor de 3 variables.
 - Converteix un caràcter numèric (char '0'..'9') a l'enter corresponent.
- A casa: Feu tots els problemes del tema 1.