@prog2ua

Tema 3: Fitxers

Programació 2

Grau en Enginyeria Informàtica
Universitat d'Alacant
Curs 2022-2023

Índex

Introducció

Què és un fitxer (1/3)

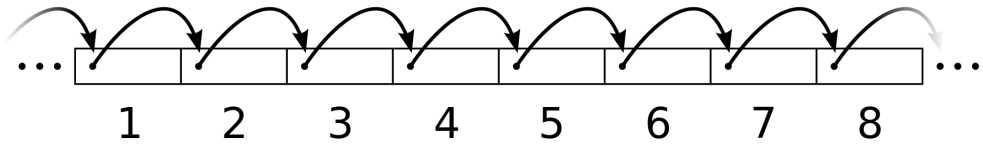
- Totes les dades amb què hem treballat fins ara s'emmagatzemen en la memòria principal de l'ordinador (*RAM*)
- La grandària de la memòria principal és prou limitada (uns pocs gigaoctets)
- Totes les dades s'esborren quan el programa acaba (*memòria volàtil*)

Què és un fitxer (2/3)

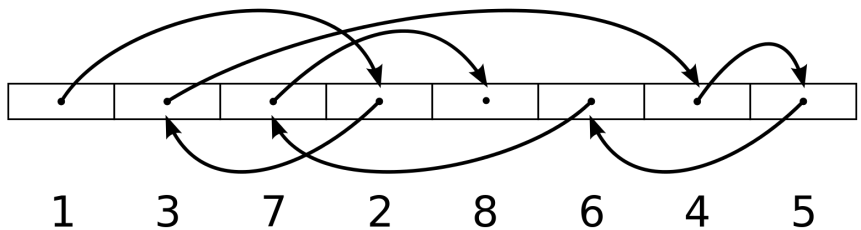
- Els *fitxers* (o *arxius*) són la forma en la qual C++ permet accedir a la informació emmagatzemada en disc (memòria secundària)
- Els fitxers són estructures dinàmiques: la grandària pot variar durant l'execució del programa segons les dades que emmagatzemen
- Existeixen dos tipus de fitxers en funció de com es guarda dins la informació: *fitxers de text* i *fitxers binaris*

Què és un fitxer (3/3)

- Hi ha dues maneres d'accedir a un fitxer:
  - *Accés seqüencial*: llegim/escrivim els elements del fitxer en ordre, començant pel principi i un darrere d'un altre



  - *Accés directe* (o *aleatori*): ens situem en qualsevol posició del fitxer i el llegim/escrivim directament, sense passar pels anteriors



Fitxers de text

Definició (1/2)

- Els fitxers de text també s'anomenen *fitxers amb format*
- Guarden la informació en forma de seqüències de caràcters, tal com es mostrarien per pantalla
- Per exemple, el valor enter `19` es guardarà en fitxer com els caràcters `1` i `9`
- Exemples de fitxers de text: un codi font en C++, una pàgina web (HTML) o un fitxer creat amb el bloc de notes
- El mode de lectura/escriptura més habitual en fitxers de text és l'accés seqüencial

Definició (2/2)

- Són fitxers que contenen solament caràcters imprimibles: aquells amb codi ASCII major o igual que 32
- El codi ASCII és un codi que assigna a cada caràcter un número per al seu emmagatzematge en memòria:

Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car
0	00	NUL	32	20	SPC	64	40	@	96	60	‘
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	’	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Declaració de variables

- Els fitxers són un tipus de dada més en C++
- Cal incloure la llibreria `fstream` en el nostre codi per a poder treballar-hi:

```
#include <fstream>
```

- Existeixen tres tipus de dades bàsiques per a treballar amb fitxers, depenent del que vulguem fer amb ells:*

```
ifstream fitxerLec; // Llegir de fitxer
ofstream fitxerEsc; // Escriure en fitxer
fstream fitxerLecEsc; // Llegir i escriure en fitxer
```

*És poc habitual usar el tipus `fstream` amb fitxers de text

Obertura i tancament (1/4)

- Una variable de tipus fitxer (*fitxer lògic*) s'ha d'associar a un fitxer real en el sistema (*fitxer físic*) per a poder llegir-hi/escriure-hi
- Per a establir aquesta relació entre la variable i el fitxer físic cal fer l'obertura del fitxer mitjançant `open`:

```
ifstream fitxer; // Llegirem del fitxer
fitxer.open("elMeuFitxer.txt");
// Ara ja podem llegir de "elMeuFitxer.txt"
```

- El nom del fitxer es pot passar com un array de caràcters o com un `string`:*

```
char nomFitxer[]="elMeuFitxer.txt";
fitxer.open(nomFitxer);
```

* Només es pot usar `string` a partir de la versió 2011 de C++

Obertura i tancament (2/4)

- A `open` se li pot passar un segon paràmetre que indica el *mode d'obertura* del fitxer:
  - Lectura: `ios::in`
  - Escriptura: `ios::out`
  - Lectura/escriptura: `ios::in | ios::out`
  - Afegir al final: `ios::out | ios::app`

```
ifstream fitxerLec;
ofstream fitxerEsc;
// Obrim només per a llegir
fitxerLec.open("elMeuFitxer.txt",ios::in);
// Obrim per a afegir informació al final
fitxerEsc.open("elMeuFitxer.txt",ios::out|ios::app);
```

Obertura i tancament (3/4)

- Si obrim un fitxer que ja existeix per a escriptura (`ios::out`) se n'esborrarà tot el contingut
- Si obrim amb `ios:app` no s'esborrarà el contingut, sinó que s'anirà afegint la nova informació al final
- Si el fitxer no existeix, es crearà un de nou amb grandària inicial `0`

Obertura i tancament (4/4)

- Per defecte, el tipus `ifstream` s'obri per a lectura i el `ofstream` per a escriptura
- Es pot obrir el fitxer en el moment de declarar-lo:

```
ifstream fl("elMeuFitxer.txt"); // Per defecte ios::in
ofstream fe("elMeuFitxer.txt"); // Per defecte ios::out
```

- Abans de llegir/escriure, s'ha de comprovar amb `is_open` si el fitxer s'ha obert correctament (`true`) o no (`false`)
- En acabar d'usar el fitxer, s'ha d'alliberar amb `close`:

```
ifstream fl("elMeuFitxer.txt");
if(fl.is_open()){
  // Ja podem treballar amb el fitxer
  ...
  fl.close();  // Tanquem el fitxer
}
else // Mostrar error d'obertura
```

Lectura amb l'operador >> (1/3)

- La lectura de fitxer permet recuperar informació guardada en disc per a posar-la en memòria i poder treballar amb ella
- Podem utilitzar l'operador `>>` per a llegir de fitxer igual que féiem amb `cin` per a llegir de teclat
- Bucle per a llegir un fitxer caràcter a caràcter:

```
ifstream fl("elMeuFitxer.txt")
if(fl.is_open()){
  char c; // Podríem llegir int, float, ...
  while(fl >> c){ // Llig mentre queden caràcters
    cout << c;
  }
  fl.close()
}
else{
  cout << "Error en obrir el fitxer" << endl;
}
```

Lectura amb l'operador >> (2/3)

- En usar l'operador `>>` descartem els blancs, igual que succeïa en llegir de `cin`
- Podem usar la funció `get` per a llegir caràcter a caràcter sense descartar blancs:

```
ifstream fl("elMeuFitxer.txt");
if(fl.is_open()){
  char c;
  while(fl.get(c)){
    cout << c;
  }
  fl.close();
}
else{
  cout << "Error en obrir el fitxer" << endl;
}
```

Lectura amb l'operador >> (3/3)

- També es pot usar l'operador >> per llegir fitxers que continguen diferents tipus de dades
- Per exemple, si tenim un fitxer que conté en cada línia una cadena i dos enters (ex. `Hola 1032 124`):

```
ifstream fl("elMeuFixter.txt");
if(fl.is_open()){
  string s;
  int num1,num2;
  while(fl >> s){ // Llig el string
    fl >> num1; // Llig el primer nombre
    fl >> num2; // Llig el segon nombre
    cout << s << "," << num1 << "," << num2 << endl;
  }
  fl.close()
}
...
```

Lectura per línies

- Podem usar la funció `getline` per a llegir una línia completa de fitxer, igual que féiem en llegir de `cin`:

```
ifstream fl("elMeuFitxer.txt");
if(fl.is_open()){
  string s;
  while(getline(fi,s)){
    cout << s << endl;
  }
  fl.close();
}
else{
  cout << "Error en obrir el fitxer" << endl;
}
```

Detecció de final de fitxer

- El mètode `eof` ens indica si s'ha arribat el final de fitxer
- Aquesta circumstància es dóna quan no queden més dades per llegir:

```
ifstream fl;
...
while(!fl.eof()){
  // Llegim utilitzant alguns dels mètodes vistos
}
```

- Quan s'intenta llegir dades que queden fora del fitxer, el mètode retorna `true`
- Després d'haver llegit l'última dada vàlida el mètode continua retornant `false`
- És necessari fer una lectura més per a provocar que `eof` retorne `true`

Escriptura amb l'operador <<

- Podem utilitzar l'operador `<<` per a escriure en fitxer igual que féiem amb `cout` per a escriure per pantalla:

```
ofstream fe("elMeuFitxer.txt");
if(fe.is_open()){
  int num=10;
  string s="Hola, mon";
  fe << "Un nombre enter: " << num << endl;
  fe << "Un string: " << s << endl;
  fe.close();
}
else{
  cout << "Error en obrir el fitxer" << endl;
}
```

Exercicis (1/6)

Exercici 1

Implementeu un programa que llija un fitxer `fitxer.txt` i imprimisca per pantalla les línies del fitxer que contenen la cadena `Hola`.

Exercicis (2/6)

Exercici 2

Feu un programa que llija un fitxer `fitxer.txt` i escriga en un altre fitxer `FITXER.TXT` el contingut del fitxer d'entrada amb totes les lletres en majúscules.

Exemple:

`fitxer.txt`	`FITXER.TXT`
`Hola, mon.`	`HOLA, MON.`
`Com estem?`	`COM ESTEM?`
`Adeu, adeu...`	`ADEU, ADEU...`

Exercicis (3/6)

Exercici 3

Feu un programa que llija dos fitxers de text, `f1.txt` i `f2.txt`, i escriga per pantalla les línies que siguen diferents en cada fitxer, amb `<` davant si la línia correspon a `f1.txt` i amb `>` si correspon a `f2.txt`.

Exemple:

f1.txt	f2.txt
hola, mon. com estem? adeu, adeu...	hola, mon. com anem? adeu, adeu...

L'eixida ha de ser:

```
< com estem?
> com anem?
```

Exercicis (4/6)

Exercici 4

Dissenyeu una funció `fiFitxer` que reba dos paràmetres: el primer ha de ser un nombre enter positiu `n` i el segon el nom d'un fitxer de text. La funció ha de mostrar per pantalla les `n` últimes línies del fitxer.

Exemple:

```
fiFitxer(3,"cadenes.txt")

with several words
unaparaula
mooooooltes paraules, moltes, moltes...
```

Exercicis (5/6)

Exercici 4 (continuació)

Hi ha dues solucions:

1. A la babalà: llegir el fitxer per a comptar les línies que té i tornar a llegir el fitxer per a escriure les `n` línies finals. Problema: i si el fitxer té 100000000000000 línies?
2. Utilitzar un array de `string` de grandària `n` que emmagatzeme en tot moment les `n` últimes línies llegides (encara que al principi tindrà menys de `n` línies)

Exercicis (6/6)

Exercici 5

Tenim dos fitxers de text, `f1.txt` i `f2.txt`, en els quals cada línia és una sèrie de nombres separats per `:`. Cada línia està ordenada pel primer nombre, de menor a major, en els dos fitxers. Feu un programa que llija els dos fitxers, línia per línia, i escriga en un fitxer `f3.txt` les línies comunes a tots dos fitxers.

Exemple:

f1.txt	f2.txt	f3.txt
10:4543:23	10:334:110	10:4543:23:334:110
15:1:234:67	12:222:222	15:1:234:67:881:44
17:188:22	15:881:44	20:111:22:454:313
20:111:22	20:454:313	

Fitxers binaris

Definició (1/2)

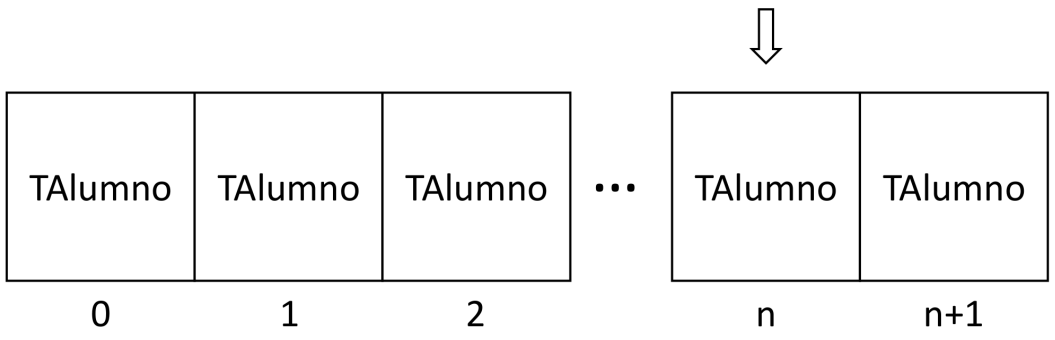
- També els anomenem *fitxers sense format*
- Guarden la informació tal com s'emmagatzema en la memòria principal de l'ordinador
- Per exemple, el valor enter (char) `19` es guardarà en fitxer com la seqüència `00010011`
- Per a llegir i escriure s'utilitzen funcions diferents a les dels fitxers de text
- Amb els fitxers binaris se sol emprar tant accés seqüencial com directe
- Les lectures i escriptures són més ràpides que amb els fitxers de text (no cal convertir a caràcter)
- Generalment ocupen menys espai en disc que els fitxers de text equivalents

Definició (2/2)

- És molt habitual que cada element del qual es vol guardar informació s'emmagatzeme en un *registre* (`struct`):

```
struct TAlumne{
  char nom[100];
  int grup;
  float notaMitjana;
};
```

- Mitjançant accés directe, es pot accedir directament a l'element `n` del fitxer sense haver de llegir els `n-1` anteriors



Declaració de variables

- Es declaren igual que en els fitxers de text:

```
#include <fstream> // Sempre que es treballa amb fitxers

ifstream fitxerLec; // Llegir de fitxer
ofstream fitxerEsc; // Escriure en fitxer
fstream fitxerLecEsc; // Llegir i escriure en fitxer
```

Obertura i tancament

- En obrir el fitxer cal indicar que és binari mitjançant el mode d'obertura `ios::binary`:
  - Lectura: `ios::in | ios::binary`
  - Escriptura: `ios::out | ios::binary`
  - Lectura/escriptura: `ios::in | ios::out | ios::binary`
  - Afegir al final: `ios::out | ios::app | ios::binary`

```
ifstream fitxerLec;
ofstream fitxerEsc;
// Obrim només per a llegir en mode binari
fitxerLec.open("elMeuFitxer.dat",ios::in | ios::binary);
// Obrim només per a escriure en mode binari
ficheroEsc.open("elMeuFitxer.dat",ios::out | ios::binary)
    ;
// Forma abreujada
fstream fitxerLecEsc("elMeuFitxer.dat",ios::binary)
```

- Igual que amb els fitxers de text, es pot comprovar si està obert amb `is_open` i es tanca amb `close`

Lectura (1/3)

- Per a llegir de fitxer binari utilitzem la funció `read`
- Aquesta funció rep dos paràmetres: el primer indica on es guardarà la informació llegida de fitxer i el segon la quantitat d'informació (nombre de bytes) que es llegirà:*

```
TAlumne alumne;
ifstream fitxer;

fitxer.open("elMeuFitxer.dat",ios::in | ios::binary);
if(fichero.is_open()){
  // En cada iteració llegim un registre TAlumne
  while (fitxer.read((char *)&alumne, sizeof(TAlumne))){
    //  Anem mostrant el nom i la nota de cada alumne
    cout << alumne.nom << ": " << alumne.nota << endl;
  }
  fitxer.close();
}
```

*Per a saber el nombre de bytes que ocupa una variable es pot usar la funció `sizeof`

Lectura (2/3)

- Es pot llegir directament un element `n` del fitxer sense haver de llegir els `n-1` anteriors (accés directe)
- La funció `seekg` permet situar-se en un punt específic del fitxer
- Rep dos paràmetres: el primer indica quants bytes ens volem saltar, mentre que el segon indica el punt de referència per a fer aquest salt

```
// Tenim un fitxer amb registres de tipus TAlumne
ifstream fitxer("elMeuFitxer.dat",ios::binary);
TAlumne alumne;
...
// Podem llegir directament el tercer registre
// Ens saltem els dos primers registres
fitxer.seekg(2*sizeof(TAlumne),ios::beg);
// Ja podem llegir el tercer registre
fitxer.read((char *)&alumne,sizeof(alumne));
...
```

Lectura (3/3)

- Punts de referència possibles:
  - `ios::beg`: comptant des del principi del fitxer
  - `ios::cur`: comptant des de la posició actual
  - `ios::end`: comptant des del final del fitxer
- Si el primer paràmetre de `seekg` és un número negatiu, la finestra de lectura es mou cap al principi del fitxer:

```
ifstream fitxer("elMeuFitxer.dat",ios::binary);
TAlumne alumne;
...
fitxer.seekg(-1*sizeof(TAlumne),ios::end);
// Llegim l'últim registre del fitxer
fitxer.read((char *)&alumne,sizeof(TAlumne));
...
```

Escriptura (1/3)

- Per a escriure en fitxers binaris utilitzem la funció `write`
- Aquesta funció rep dos paràmetres: el primer indica on està guardada la informació que volem escriure i el segon la quantitat d'informació (nombre de bytes) que s'escriuran
- La sintaxi és molt semblant a la de `read`:

```
ofstream fitxer("elMeuFitxer.dat", ios::binary);
TAlumne alumne;

if(fichero.is_open())
{
  strcpy(alumne.nom,"Pepe Pi");
  alumne.notaMitjana=7.8;
  alumne.grup=5;

  fitxer.write((const char *)&alumne, sizeof(TAlumne));
  fitxer.close();
}
```

Escriptura (2/3)

- Igual que per a la lectura, es pot escriure directament en el registre `n` sense haver d'escriure els `n-1` anteriors
- La funció `seekp` permet posicionar-se per a escriptura (`seekg` és per a lectura)
- Els paràmetres són els mateixos que per a `seekg`:

```
ofstream fitxer("elMeuFitxer.dat",ios::binary);
TAlumne alumne;
...
// Ens posicionem per a escriure el tercer registe
fitxer.seekp(2*sizeof(TAlumne),ios::beg);
fitxer.write((const char *)&alumne,sizeof(TAlumne));
...
```

- Si la posició a la qual anem amb `seekp` no existeix en el fitxer, aquest s'"allarga" perquè s'hi puga escriure

Escriptura (3/3)

- Per a emmagatzemar cadenes de caràcters en un fitxer binari s'han d'usar arrays de caràcters, mai `string`
- El problema de `string` és que és una dada de grandària variable, per la qual cosa no podem tindre registres que tinguen tots la mateixa grandària
- Pot ser necessari retallar la cadena perquè càpia en el registre abans de guardar-lo en fitxer:

```
const int TAM=20;
char cad[TAM];
string s;
...
strncpy(cad,s.c_str(),TAM-1); // Màxim 19 caràcters
cad[TAM-1]='\0';
```

Posició actual

- La posició actual (en bytes) de la finestra de lectura es pot obtindre mitjançant la funció `tellg` i la d'escriptura mitjançant `tellp`
- Es pot usar, per exemple, per a calcular el nombre de registres d'un fitxer:

```
ifstream fitxer("elMeuFitxer.dat",ios::binary);
// Col·loquem la finestra de lectura al final del fitxer
fitxer.seekg(0,ios::end);
// Calculem el nombre de registres TAlumne del fitxer
cout << fitxer.tellg()/sizeof(TAlumne) << endl;
```

Exercicis (1/3)

Exercici 6

Tenim un fitxer binari `alumnes.dat` que té registres d'alumnes amb la següent informació:

- `dni`: array de 10 caràcters
- `cognoms`: array de 40 caràcters
- `nom`: array de 20 caràcters
- `grup`: enter

Feu un programa que imprimisca per pantalla el DNI de tots els alumnes del grup 7.

Ampliació: feu un programa que intercanvie els alumnes dels grups 4 i 8 (els grups van de l'1 al 10).

Exercicis (2/3)

Exercici 7

Donat el fitxer `alumnes.dat` de l'exercici anterior, feu un programa que passe a majúscules el nom i els cognoms del cinqué alumne del fitxer, tornant-lo a escriure.

Exercici 8

Dissenyeu un programa que cree el fitxer `alumnes.dat` a partir d'un fitxer de text `alu.txt` en el qual cada dada (dni, cognoms, etc.) està en una línia diferent. Tingueu en compte que en el fitxer de text el dni, nom i cognoms poden ser més llargs que les grandàries especificades per al fitxer binari, i en aquest cas s'hauran de retallar.

Exercicis (3/3)

Exercici 9

Escriviu un programa que s'encarregue de l'assignació automàtica d'alumnes en 10 grups de pràctiques. A cada alumne se li assignarà el grup corresponent a l'últim número del seu DNI (als alumnes amb DNI acabat en 0 se'ls assignarà el grup 10). Les dades dels alumnes estan en un fitxer `alumnes.dat` amb la mateixa estructura que en els exercicis anteriors.

L'assignació de grups ha de fer-se llegint el fitxer una sola vegada i sense emmagatzemar-lo en memòria. En cada pas es llegirà la informació corresponent a un alumne, es calcularà el grup que li correspon i es guardarà el registre en la mateixa posició.