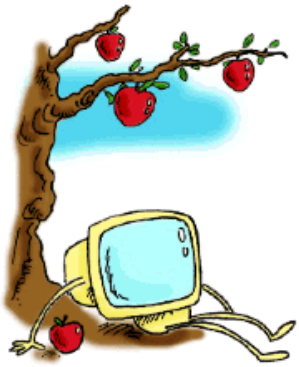
Tema 8: L'apuntador



8.1 Introducció a l'apuntador (I)

- Recordant el que s'ha dit sobre l'apuntador:
  - L'apuntador no és res més que una variable que emmagatzema l'adreça d'una altra variable.Amb l'apuntador es pot:
    - Referenciar l'adreça que emmagatzema l'adreça d'una altra variable. Ho fa l'adreça de la variable apuntador.
    - Referenciar la variable que conté la dada. Ho fa la variable apuntador.
    - Referencar el contingut de l'adreça de la variable continguda en l'adreça de l'apuntador. Ho fa la indirecció de l'apuntador.
  - Es tenen, per tant, dos operadors d'adreçament amb els que es treballarà.

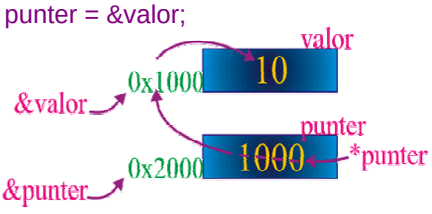
& → operador de direcció.

* → operador d'indirecció.

&valor val 1000	//la variable valor es troba en l'adreça 1000
valor val 10;	//contingut de la variable valor: <valor> = 10
&punter val 2000;	// la variable punter es troba en l'adreça 2000
punter val 1000;	//contingut de la variable punter: <punter>=1000 (& valor)
*punter val 10;	//contingut de la variable emmagatzemada en *punter:* <<punter>> = *(&valor)

Si ara es fa *punter = 20; la variable valor canvia a: → valor = 20

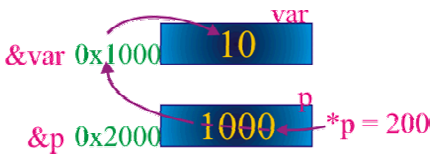
int valor = 10;
int *punter;
punter = &valor;



8.1 Introducció a l'apuntador (II)

- Quan es treballa amb apuntadors s'ha de vigilar el contingut de les variables a la que referencien. Exemple:

```
char *p;
char var:
var = 100;
p = &var;
*p = 200;          // Això implica que var = 200.
```



- **Declaració d'apuntadors.**
- Quan es declara un apuntador s'ha d'especificar el tipus de variable a la que s'apunta. Exemple:

```
int *num;          → num apuntarà a una variable tipus enter.
char *car[5];      → car apuntarpà a un cadena.
```

- **Inicialització d'apuntadors.**
- Els apuntadors es poden inicialitzar en declarar-se.
- Sempre ha de coincidir el tipus de variable a la que apunta amb el valor a inicialitzar.
- Exemple:

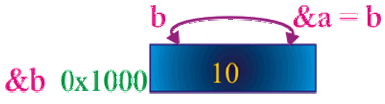
```
char *car = "Hola";
int *dia = &dies[0];    //S'apunta a la primera adreça d'un array d'enters
int *dia = dies;        // S'apunta a la primera adreça d'un array d'enters (com el cas anterior)
```


8.1 Introducció a l'apuntador (III)

- **Variables de referència**
  - S'ha d'anar amb compta quan es treballa amb variables de referència. Els dos casos següents fan el mateix de forma diferent:

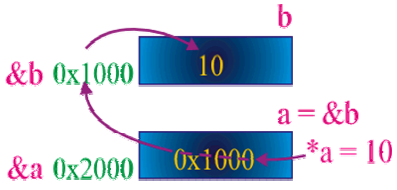
1. Aquí es treballa amb una mateixa posició de memòria. És el cas del pas de paràmetres per referència.

```
float b;
float &a = b; //L'adreça d'a és la de la variable b.
a = 10;        //(→ b =10)  (a i b són la mateixa variable)
```



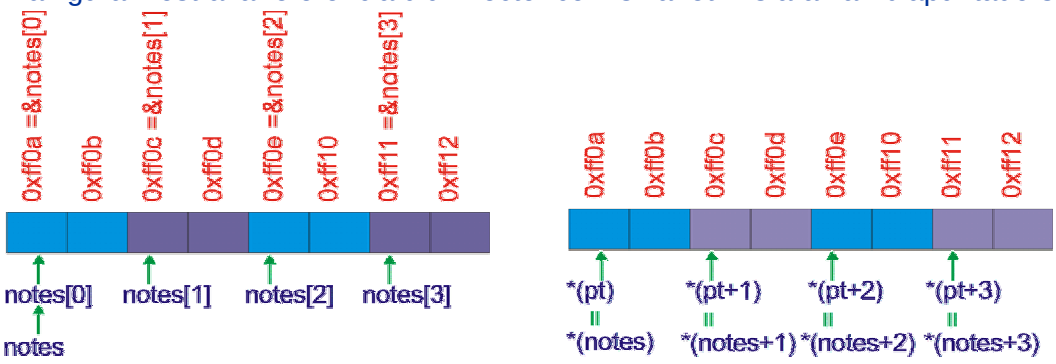
2. Ara es treballa amb els continguts de les posicions de memòria. És el cas del pas de paràmetres per apuntador.

```
float b;
float * a = &b;
*a = 10;       //(→ b=10) (es treballa amb els continguts)
```



8.2 L'apuntador en arrays (I)

- Els diferents elements d'un array s'emmagatzemen en memòria consecutivament.
- La relació entre apuntador i nom de l'array és directa, ja que aquest indica l'adreça de començament de l'array.
- La figura mostra la referència d'un vector com s'ha fet fins ara i amb apuntadors.



- Un cop declarat l'array, la referenciació amb apuntadors ve donada per:
  - int * pt;
    pt = &notes[0]; (ò pt = notes; → notes no és res més que una constant a una adreça!
    → compta per què pt=&notes; **és incorrecte**)
  - També es pot definir i declarar al mateix temps
    int * pt = &notes[0]; (ò int * pt = notes;) → És incorrecte
- Aleshores les adreces es poden obtenir immediatament a partir de:
  &notes[i] = &notes[i] + i*sizeof(int);

8.2 L'apuntador en arrays (II)

- Quan s'usen apuntadors per a tractar arrays, s'ha de tenir en compta que en realitzat es manegen posicions de memòria → S'ha d'emprar *(m+i) enlloc de m[i] !
- Exemple: Càlcul del número màxim en un array d'enters

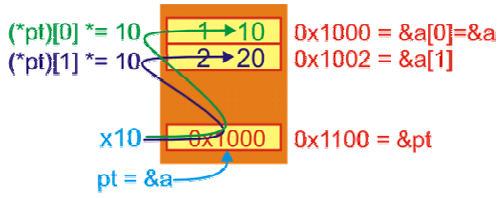
```
//Cas 1: Treballant com a vectors en la funció
#include <stdio.h>
#define N=6;
int maxim (int *);      //Declaració funció
void main()
{
    int a[]={3,8,5,1,7,6};
    printf("Maxim= %i \n", maxim(a) );
}
int maxim (int *m)      //Definició funció
{
    int i, max=m[0];
    for (i=0; i<N; i++)
        if (max<m[i]) max = m[i];
    return (max);
}
```

```
//Cas 2: Treballant amb apuntadors en la funció
#include < stdio.h>
#define N=6;
int maxim (int *);      //Declaració funció
void main()
{
    int a[]={3,8,5,1,7,6};
    printf("Maxim= %i \n", maxim(a) );
}
int maxim (int *m)      //Definició funció
{
    int i, max=*m;
    for (i=0; i<N; i++)
        if (max<*(m+i)) max = *(m+i);
    return (max);
}
```

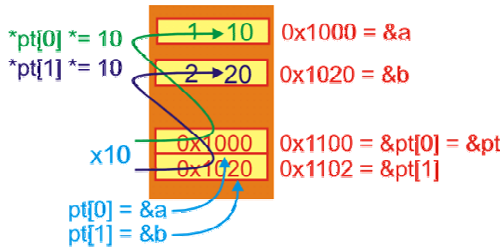
8.3 Arrays d'apuntadors

- És recomanable anar pas a pas quan es treballa amb apuntadors. *D'apuntar a un array a tenir un array d'apuntadors només hi va un parèntesi!*

```
//Cas 1: Treballant com a vectors en la funció
//int (*ptr)[ ] → Apuntador a un array d'enters
//fòrmula més complicada envers la mostrada
//prèviament, però que il·lustra millor l'exemple!
#include <stdio.h>
void main()
{
    int i, a[2]={1, 2};
    int (*pt)[2];              //Millor int *pt;
    pt=&a;                     //Millor pt=a; o només a;
    for (i=0;i<2;i++) (*pt)[i] *= 10;      //Seria *(pt+i)
    for (i=0;i<2;i++) printf("%i ", a[i]); //→ 10, 20
}
```



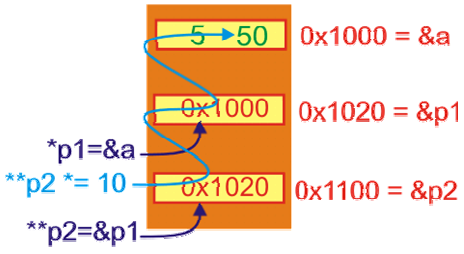

```
//Cas 2: Treballant amb apuntadors en la funció
//int *ptr[ ] → Array d'apuntadors a enters
#include <stdio.h>
void main()
{
    int i, a=1, b=2;
    int *pt[2];
    pt[0]=&a;
    pt[1]=&b;
    for (i=0;i<2;i++)  *pt[i] *= 10;
    for (i=0;i<2;i++) printf("%i ", *pt[i]); // → 10,20
}
```



8.4 Apuntadors d'apuntadors

- Com el seu nom indica són apuntadors que apunten a apuntadors.
  - Exemple:

```
#include <stdio.h>
void main()
{
    int a=5;
    int *p1=&a;
    int **p2=&p1;              //apunta a p1 que apunta a --> **p2=<a>=5
    printf("%i\n", **p2);      //imprimeix 5
    **p2 *= 10;
    printf("%i\n", a);         //imprimeix 50
}
```



8.5 L'apuntador en el pas de paràmetres (I)

- Tal com s'ha introduït en el capítol 5, l'apuntador és útil en el pas de paràmetres.
- Així, mentre el pas de paràmetres per referència realitza el pas de paràmetres passant l'adreça, en el pas de paràmetres per apuntador es passa, realment, el contingut de les posicions de memòria.
- Exemples comparatius: intercanvi dels valors de dues variables.

```
//Intercanvi de variables: pas per adreça
#include <stdio.h>
void canvi(int &, int &);          //Declaració
void main()
{
        int a=10, b=20;
        canvi (a, b);              //Crida
        printf("a=%i, b=%i\n",a b);
}
void canvi(int &x, int &y)         //Definició
{
        int temp;
        temp = x;
        x = y;
        y = temp;
}
```

```
//Intercanvi de variables: pas per contingut
#include < stdio.h>
void canvi(int *, int *);          //Declaració
void main()
{
        int a=10, b=20;;
        canvi (&a, &b);            //Crida
        printf("a=%i, b=%i\n",a b);
}
void canvi(int *x, int *y)         //Definició
{
        int temp;
        temp = *x;
        *x = *y;
        *y = temp;
}
```


8.5 L'apuntador en el pas de paràmetres (II)

- El pas de paràmetres es força distret en C. Per això val la pena entendre bé quins són els resultats en els següents casos:

```
#include <stdio.h>

int proc(int);
void main()
{
    int x = 5, a = 0;
    printf("x=%i,a=%i\n",x,a);
    a = proc(x);
    printf("x=%i,a=%i\n",x,a);
}

int proc(int p)
{
    p += 20;
    return(p);
}
```

x=5,y=0
x=5,y=25

```
#include <stdio.h>

int proc(int&);
void main()
{
    int x = 5, a = 0;
    printf("x=%i,a=%i\n",x,a);
    a = proc(x);
    printf("x=%i,a=%i\n",x,a);
}

int proc(int& p)
{
    p += 20;
    return(p);
}
```

x=5,y=0
x=25,y=25

```
#include <stdio.h>

int proc(int*);
void main()
{
    int x = 5, a = 0;
    printf("x=%i,a=%i\n",x,a);
    a = proc(&x);
    printf("x=%i,a=%i\n",x,a);
}

int proc(int* p)
{
    *p += 20;
    return(*p);
}
```

x=5,y=0
x=25,y=25

```
#include <stdio.h>

int proc(int*);
void main()
{
    int x = 5, a = 0;
    printf("x=%i,a=%i\n",x,a);
    a = proc(&x);
    printf("x=%i,a=%i\n",x,a);
}

int proc(int* p)
{
    int q = *p;
    q += 20;
    return(q);
}
```

x=5,y=0
x=5,y=25

8.6 L'apuntador en estructures

- En aquest apartat es recull l'ús d'apuntadors en estructures amb l'exemple introduït ja en el capítol anterior.
- És convenient recordar que:
  - Mitjançant un apuntador: float codi(data *); en la declaració i la definició es recullen l'adreça de la posició de memòria passada en la crida a la funció.
  - S'ha de vigilar l'ús del punter en l'estructura, ja que s'ha d'apuntar a l'adreça on es defineix l'estructura. Així (*d).dia apunta a un tipus data, mentre que *d.dia apunta a un enter!
  - Notació apuntador->membre. Com que usar apuntadors d'aquesta forma és molt comú, se sol usar l'operador ->. Això, les crides del tipus (*d).dia se simplifiquen a d->dia , tal com mostra la columna de la dreta.

```
//Estructura
struct data
{
   int dia;
   int mes;
   int any;
};
```

```
data calcular(data*); //Funció
void main()  //Progr. principal
{
   data d2, d1 = {21, 2, 2001};
   imprimir(d1, 1);
   d2=calcular(&d1);
   imprimir(d1, 2);
   imprimir(d2, 3);
}
```

```
//Funció calcular
//Amb operador *
data calcular(data *d)
{
    (*d).dia  = (*d).dia+1;
    (*d).mes = (*d).mes+1;
    (*d).any  = (*d).any+1;
   return (*d);
}
```

```
//Funció calcular
//Amb operador ->
data calcular(data *d)
{
    d->dia   = d->dia+1;
    d->mes = d->mes+1;
    d->any  = d->any+1;
   return (*d);
}
```


8.7 Aritmètica d'apuntadors (I)

- S'ha de tenir present que l'apuntador emmagatzema una adreça. Per tant, amb els apuntadors es poden realitzar totes les operacions aritmètiques i relacionals que s'han vist ... amb compta perquè els operands ara són adreces!
- Per tant, en sumar i restar números a apuntadors es poden obtenir diferentes adreces (tal com es feia amb els arrays).
- Cal vigilar, però, l'ordre en què es fan les operacions.
- Les operacions que es realitzen amb els operadors increment/decrement són:
  - Sigui pt un apuntador a un array (per exemple, de caràcters). Aleshores
    - *pt++ → S'usa l'apuntador i després s'incrementa → Passa a apuntar al següent element de l'array.
    - *--pt → Es passa a apuntar a l'element anterior de l'array i després s'usa.
    - (*pt)++ → És un cas diferent: Incrementa el valor de la variable a la que s'està referenciant.
- En registres l'operador -> té prioritat superior respecte als operadors ++/--.
  - Sigui, ara, pt un apuntador a un array de registres tipus dia:
    - ++pt->dia → Primer s'accedeix a dia i després s'aplica l'increment.
    - (++pt)->dia → S'incrementa pt passant a la següent adreça i després s'accedeix a dia.
    - (pt++)->dia → Primer s'accedeix a dia i després s'empra l'operador increment en pt

8.7 Aritmètica d'apuntadors (II)

- Per tant, s'ha d'anar molt en compta. Observem els següents exemples:
  - Sigui la següent cadena i l'apuntador que l'apunta:

```
char cad[]="Frase de prova";
char *p;
p=&cad[0];
```

que es llegeix amb el programa següent:

```
while (*p != '\0')
{
    printf("%c",*p);
    p++;
}
```

o que es pot resumir amb la instrucció següent:

```
while (*p != '\0') printf("%c",*p++);
```


8.8 Exemples (I)

- Programa que fa la mitja aritmètica d'un array emprant apuntadors.

```
#include <stdio.h>
#define DIM 6
void main()
{
    float a[DIM]={3, 4.5, 10, 7.3, 9, 12.7};
    float sum=0;
    for (int i=0;i<DIM; i++) sum +=*(a+i);
    printf("Mitja aritmètica = %2.1f\n", sum/DIM);
}
```

- Programa que compta el nombre d'a's que apareixen en una frase.

```
#include <stdio.h>
#define LEN 80
void main()
{
    char cad[LEN];
    int n=0;
    printf("Entra cadena: ");
    gets(cad);
    for (int i=0;i<LEN; i++) if (*(cad+i)=='a') n++;
    printf("Nombre d'a's = %2.1i\n", n);
}
```