INFORMÁTICA INDUSTRIAL

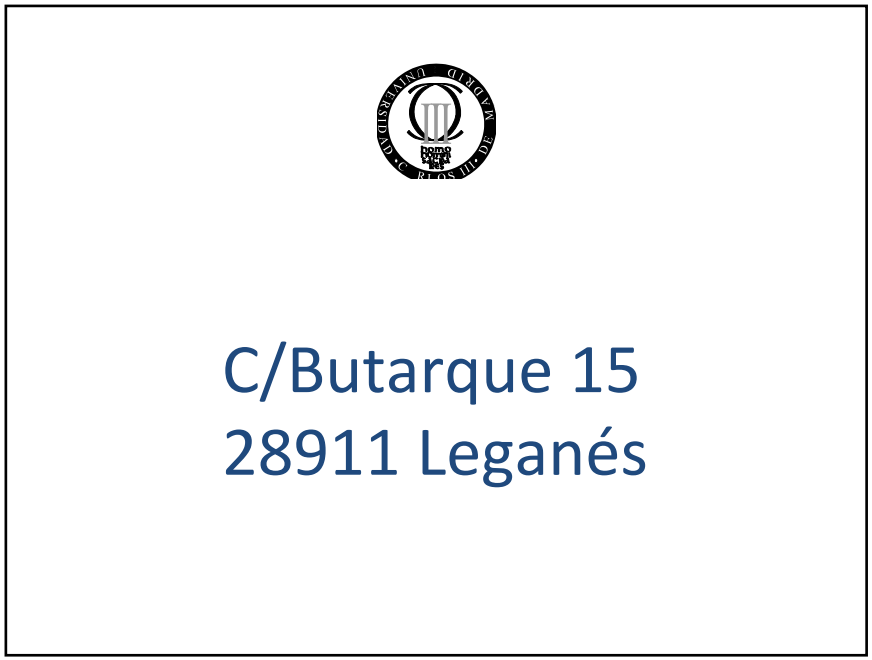
PROGRAMACIÓN BÁSICA C++ (III)

M. Abderrahim, A. Castro, J. C. Castillo
Departamento de Ingeniería de Sistemas y Automática

uc3m | Universidad Carlos III de Madrid

8. Punteros

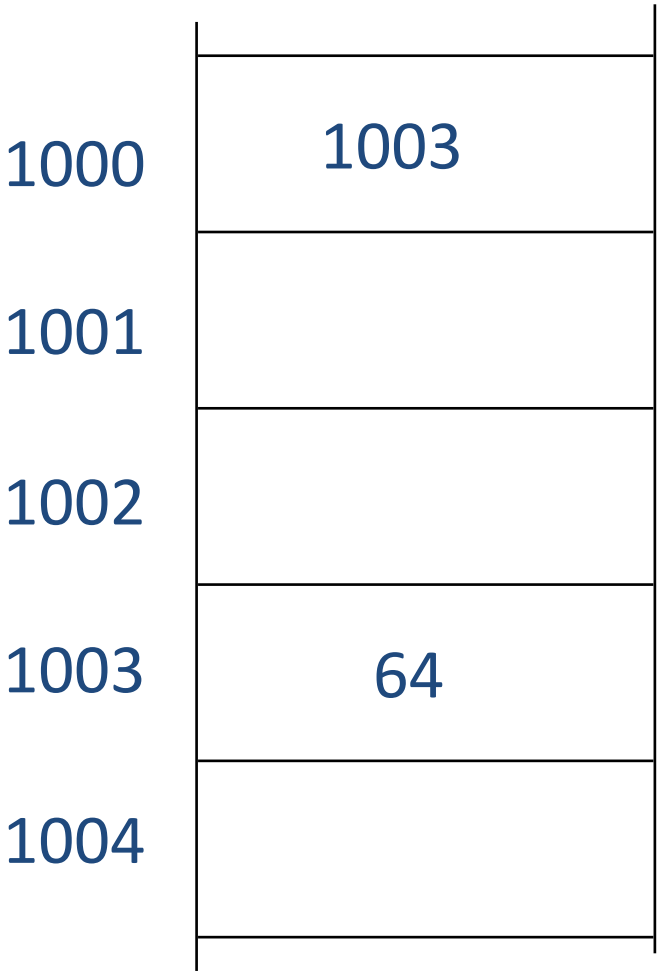
Punteros



Nombre de la variable

Dirección de la variable

Punteros




```
int a = 64;
int * p = &a;
```

- *a* es una variable que vale 64
- *p* es una variable que vale la dirección de memoria de otra variable

Punteros

- Es una variable que señala a otra variable o función
- Declaración:

```
tipo* nombre_del puntero;
int* p;
```

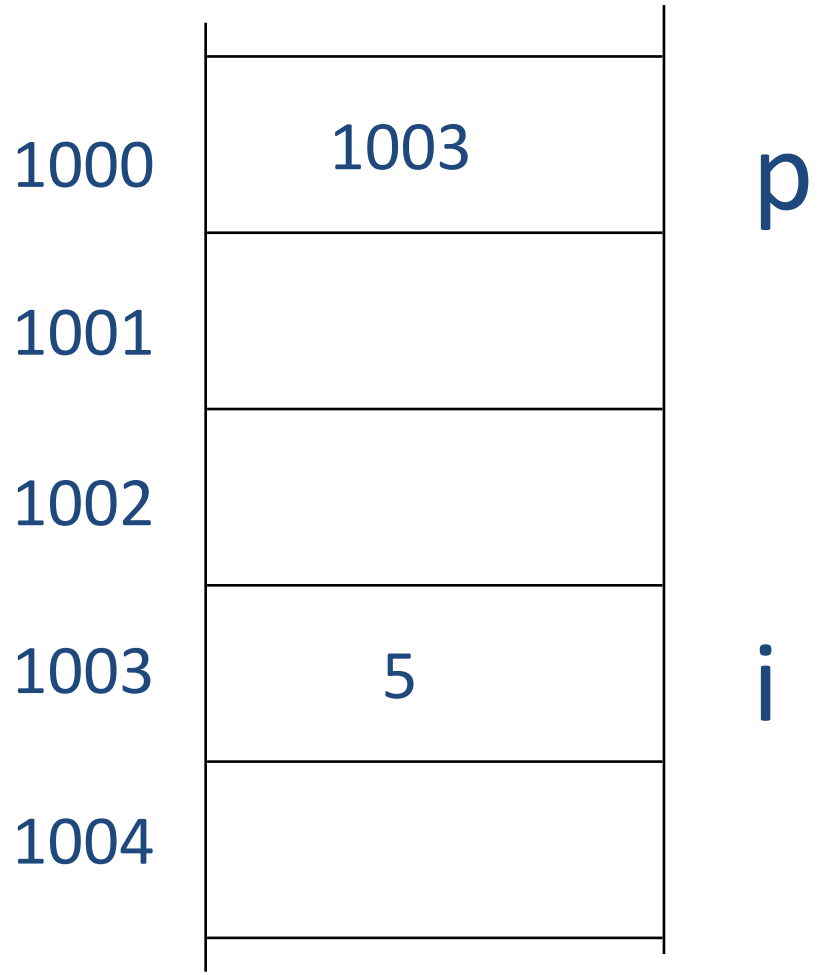
- p es un puntero a una variable de tipo entero.
- p NO es un entero

Punteros

- **&** se usa para obtener la dirección de variables
- * se usa para obtener el valor que hay en una dirección de memoria

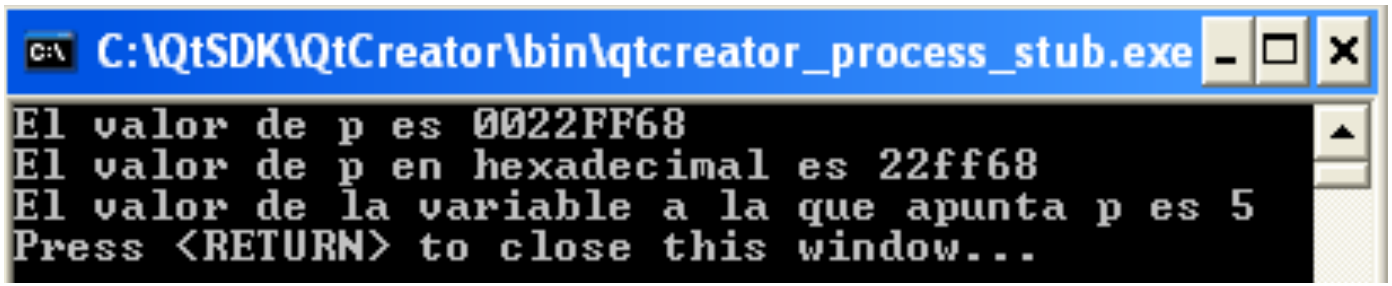
Inicialización de punteros

```
int i;
int *p;
i=5;
p=&i;
```



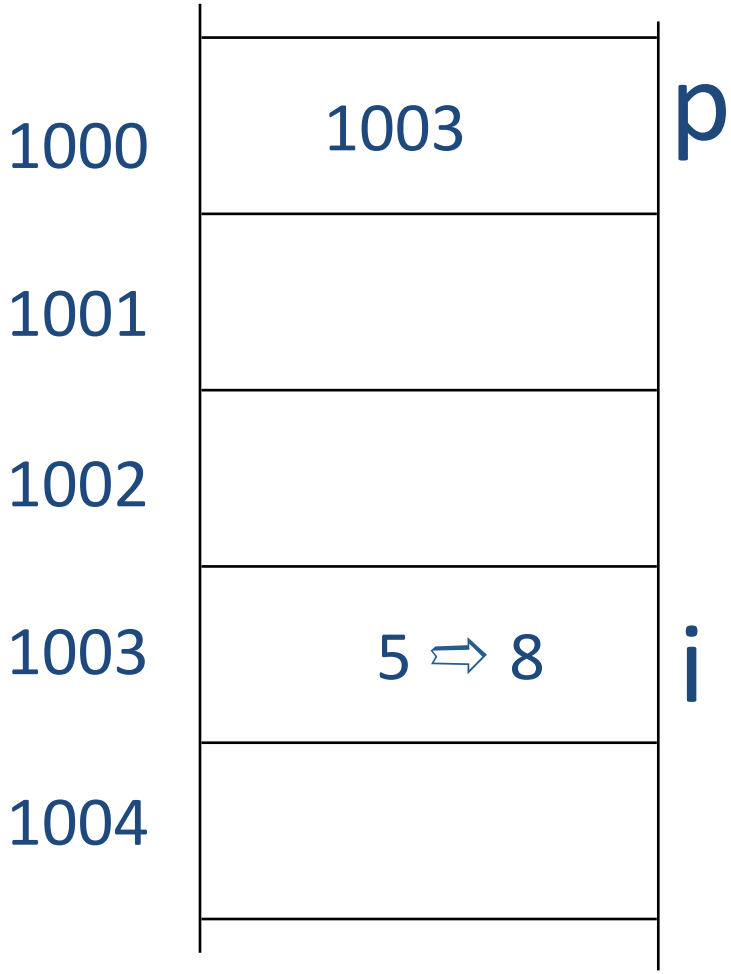
Inicialización de punteros

```
#include <iostream>
using namespace std;
int main(){
	int i = 5;
	int *p;
	p = &i;
	cout<<"El valor de p es "<< p << endl;
	cout<<"El valor de p en hexadecimal es "<< hex << p << endl;
	cout<<"El valor de la variable a la que  apunta p es "<< *p << endl;
	return 0;
}
```



Acceso a variables por punteros

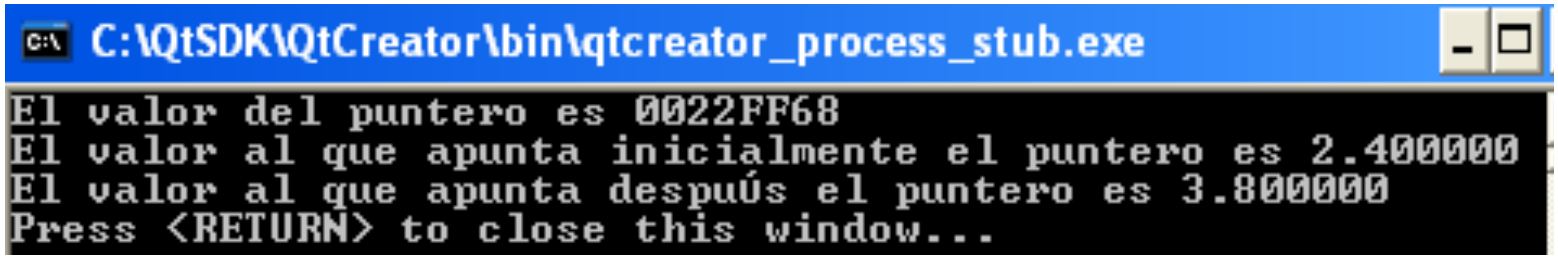
```
int i;
int *p;
i=5;
p=&i;


i=8;
```




```
int i;
int *p;
i=5;
p=&i;


*p=8;
```

Acceso a variables por punteros

```
#include <iostream>
using namespace std;
int main(){
	float f ;
	float *p;
	f = 2.4;
	p = &f;
	cout<<"El valor del puntero es "<< p << endl;
	cout<<"El valor al que apunta inicialmente el puntero es "<< *p << endl;
	f = 3.8;
	cout<<"El valor al que apunta después el puntero  es "<< *p << endl;
	return 0;
}
```

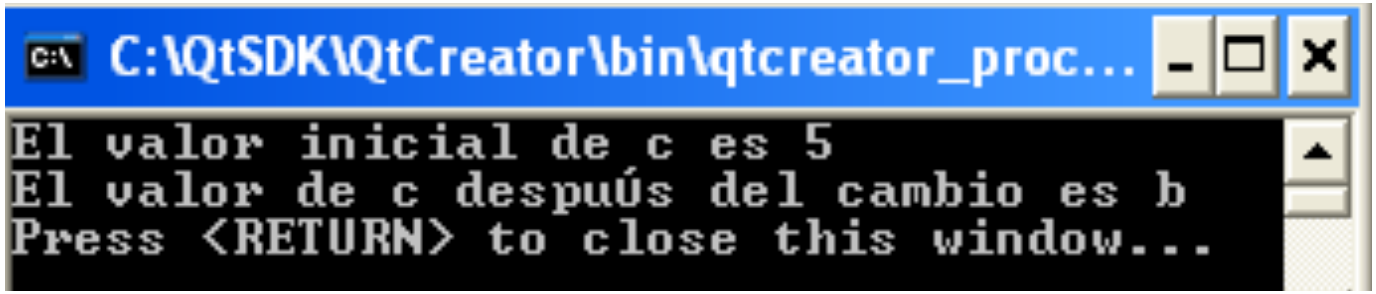


Acceso a variables por punteros

```
#include <iostream>
using namespace std;
int main(){
	char c ;
	char *p;
	p = &c;
	*p = '5'
	cout<<"El valor inicial de c es "<< c << endl;
	*p = ‘b';
	cout<<" El valor de c después del cambio es "<< c << endl;
	return 0;
}
```

Acceso a variables por punteros

```
#include <iostream>
using namespace std;
int main(){
	char c ;
	char *p;
	p = &c;
	*p = '5'
	cout<<"El valor inicial de c es "<< c << endl;
	*p = ‘b';
	cout<<" El valor de c después del cambio es "<< c << endl;
	return 0;
}
```



Punteros y funciones

- Si se quiere modificar el valor de una variable dentro de una función esta debía ser global, por lo que se pierde el control del programa.
- Se puede usar referencia.
- Ahora tenemos la opción de hacerlo pasando la dirección de la variable en lugar del valor o referencia.

Punteros y funciones

```
main()
{
    int x1 = 100, x2 = 200;
    cout<< x1 << x2;
    exchange(x1, x2);
    cout<< x1<< x2;
}
```

```
void exchange(int n1,int n2)
{
    int temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
}
```

Se imprimirá:

Punteros y funciones

```
main()
{
    int x1 = 100, x2 = 200;
    cout<< x1 << x2;
    exchange(x1, x2);
    cout<< x1<< x2;
}
```

```
void exchange(int n1,int n2)
{
    int temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
}
```

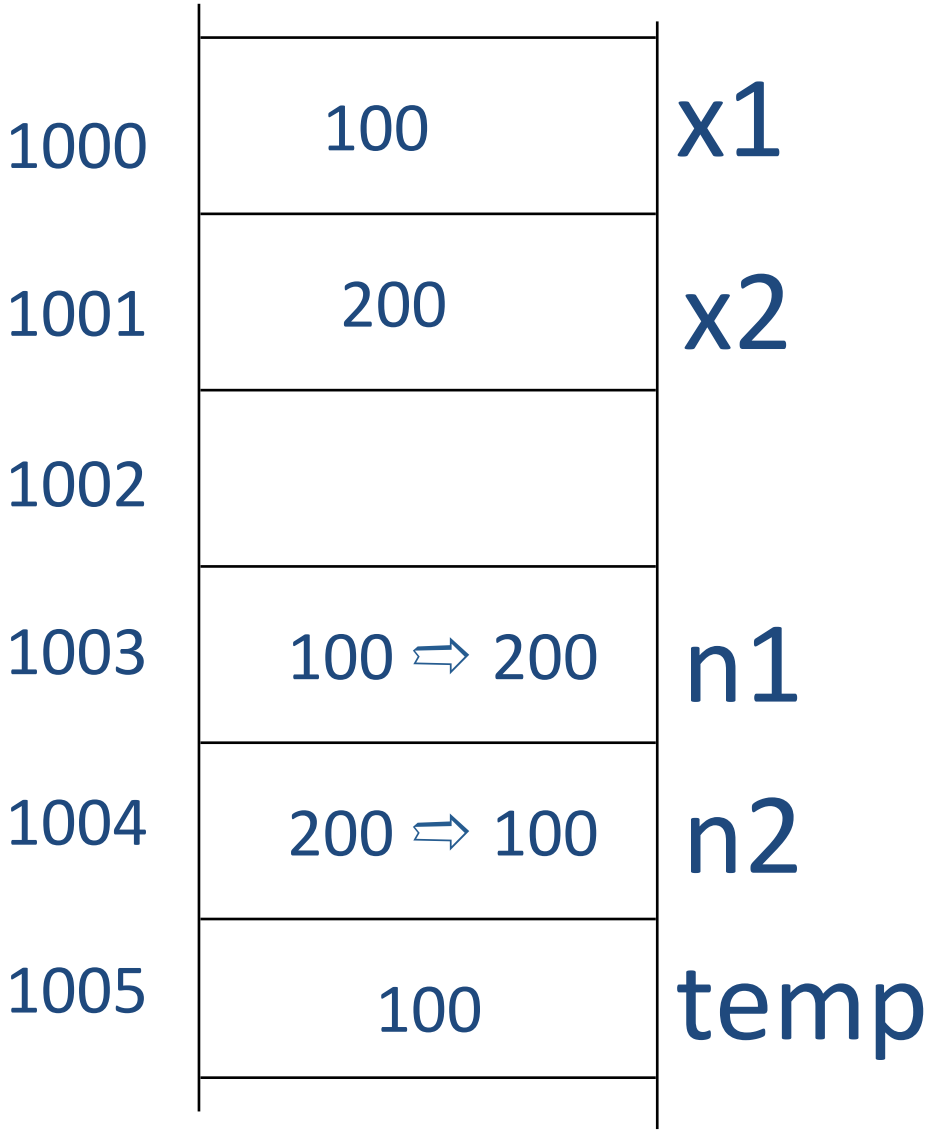
Se imprimirá: **100 200**

100 200

n1=200 x1=100
n2=100 x2=200

¿Es correcto el formato imprimido?

Punteros y funciones




```
cambiar(x1,x2);

void cambiar(int n1,int n2)
{
    int temp;
    temp=n1;
    n1=n2;
    n2=temp;
}
```

Punteros y funciones

```
main()
{
    int x1 = 100, x2 = 200;
    cout<< x1<< " " << x2 <<endl;
    exchange(x1, x2);
    cout<< x1 << " "<< x2;
}
```

```
void exchange(int *n1,int *n2)
{
    int temp;
    temp = *n1;
    *n1 = *n2;
    *n2 = temp;
}
```

Se imprimirá:

200 10

Punteros y funciones

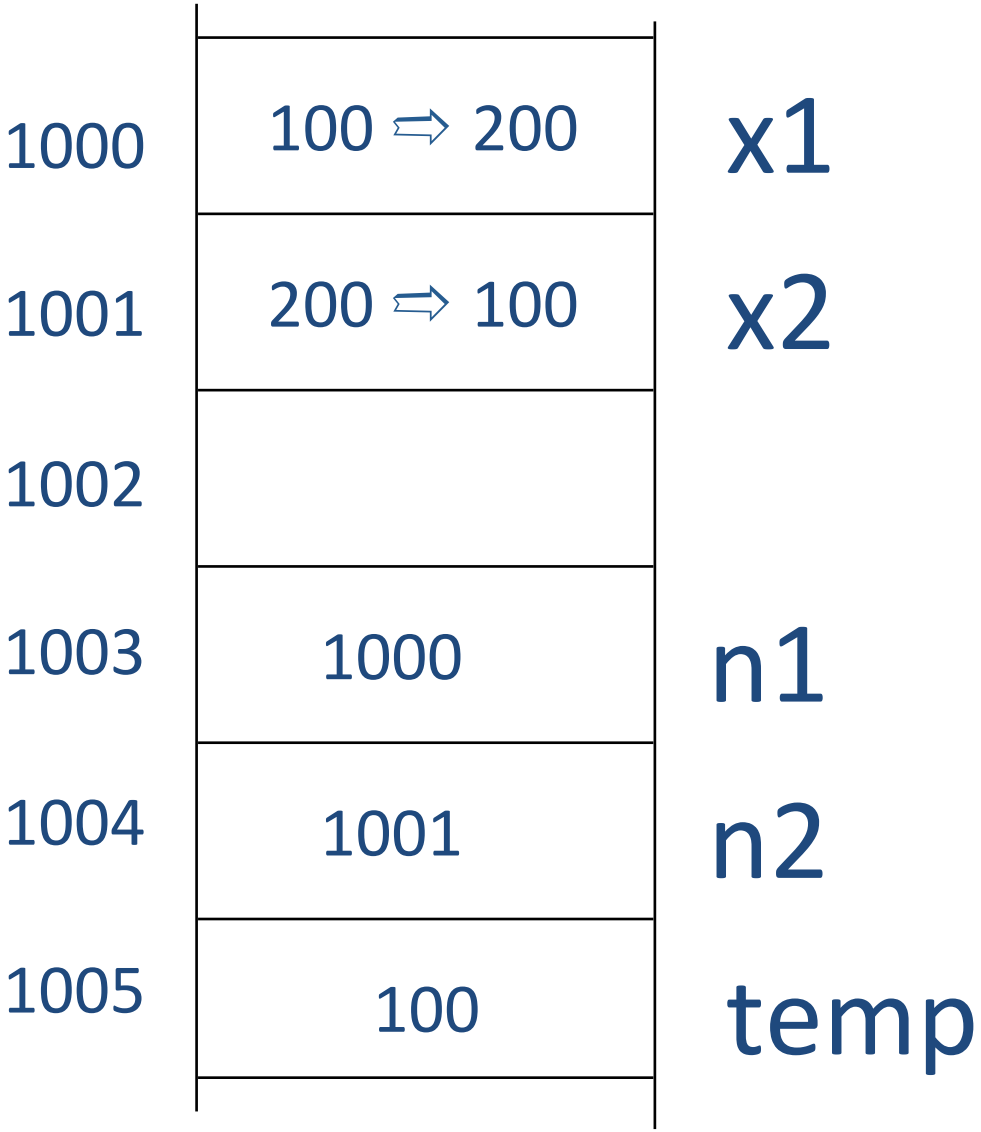
```
main()
{
    int x1 = 100, x2 = 200;
    cout<< x1<< " " << x2 <<endl;
    exchange(x1, x2);
    cout<< x1 << " "<< x2;
}
```

```
void exchange(int *n1,int *n2)
{
    int temp;
    temp = *n1;
    *n1 = *n2;
    *n2 = temp;
}
```

Se imprimirá: **100 200**

200 100

Punteros y funciones




```
cambiar(&x1,&x2);

void cambiar(int *n1,int *n2)
{
    int temp;
    temp=*n1;
    *n1=*n2;
    *n2=temp;
}
```

Punteros y funciones

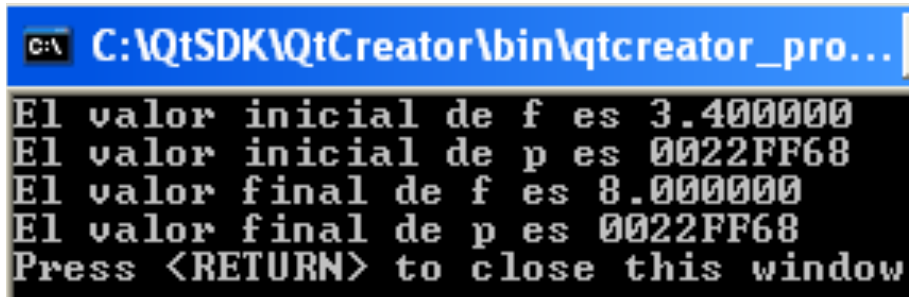
- Se crea en la función dos punteros
- Estos punteros, aunque no sean de main, apuntan a las variables de main
- Por tanto, si modificamos a lo que apuntan los punteros, estamos modificando las variables de main.

Punteros y funciones

```
#include <iostream>
using namespace std;
void FijaAOcho(float *pun);

int main(){
	float f = 3.4;
	float *p;
	p = &f;
	cout<<"El valor inicial de f es "<< f << endl;
	cout<<"El valor inicial de p es "<< p << endl;
	FijaAOcho(p);
	cout<<"El valor final de f es "<< f << endl;
	cout<<"El valor final de p es "<< p << endl;
	return 0;
}

void FijaAOcho(float *pun){
	*pun = 8;
	pun = 0;
}
```

Punteros y funciones

```
#include <iostream>
using namespace std;
void FijaAOcho(float *pun);

int main(){
	float f = 3.4;
	float *p;
	p = &f;
	cout<<"El valor inicial de f es "<< f << endl;
	cout<<"El valor inicial de p es "<< p << endl;
	FijaAOcho(p);
	cout<<"El valor final de f es "<< f << endl;
	cout<<"El valor final de p es "<< p << endl;
	return 0;
}

void FijaAOcho(float *pun){
	*pun = 8;
	pun = 0;
}
```



Punteros y arrays

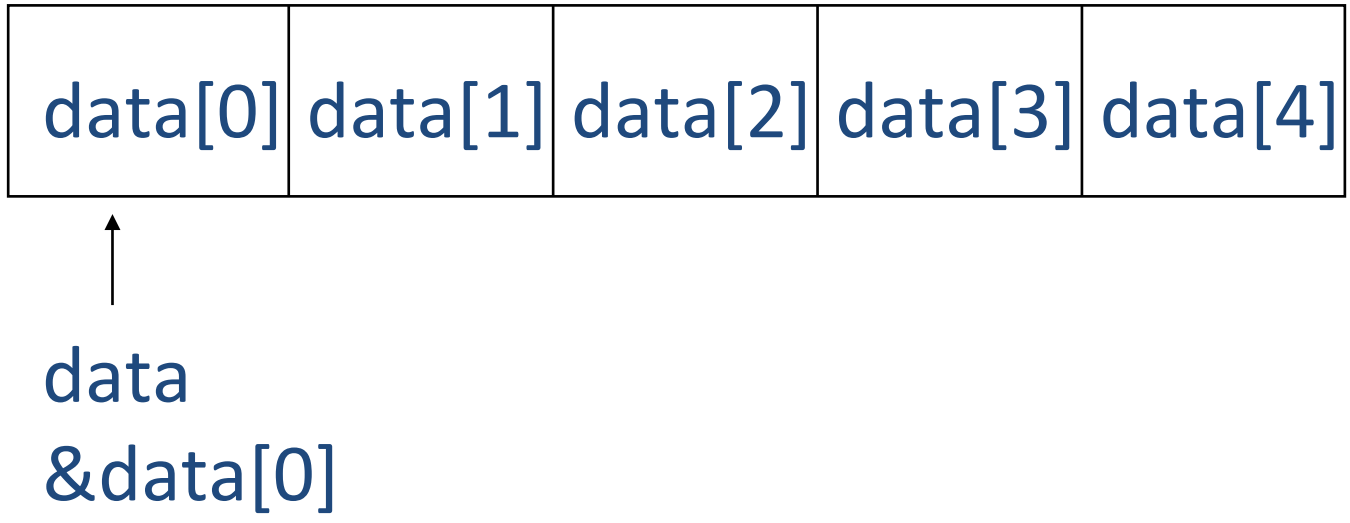
- El nombre de un array es un puntero al primer elemento del array

```
char data[5],*p;

p=&data[0];   es equivalente a     p=data;
```

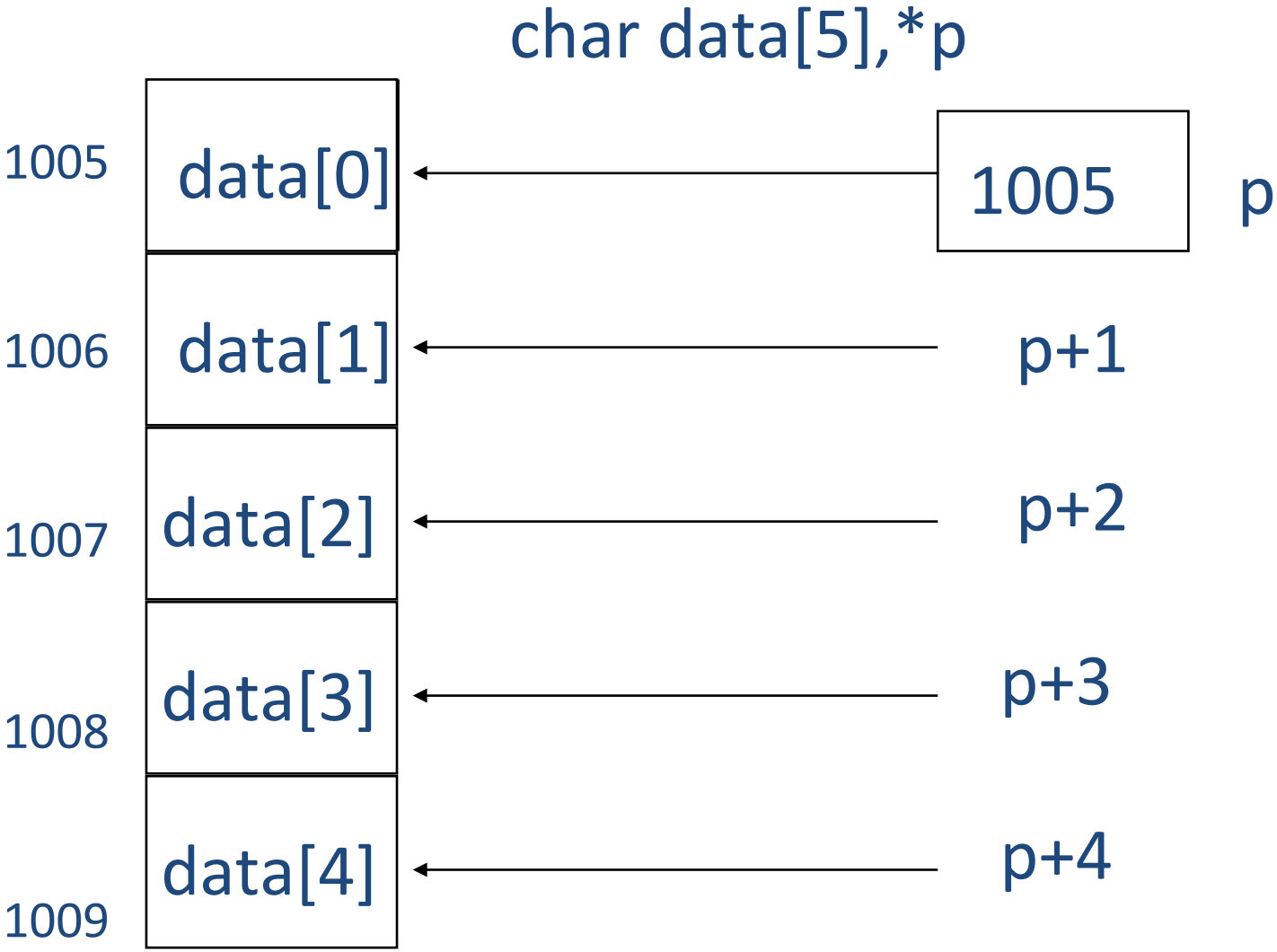
Punteros y arrays

```
char data[5],*p;
```

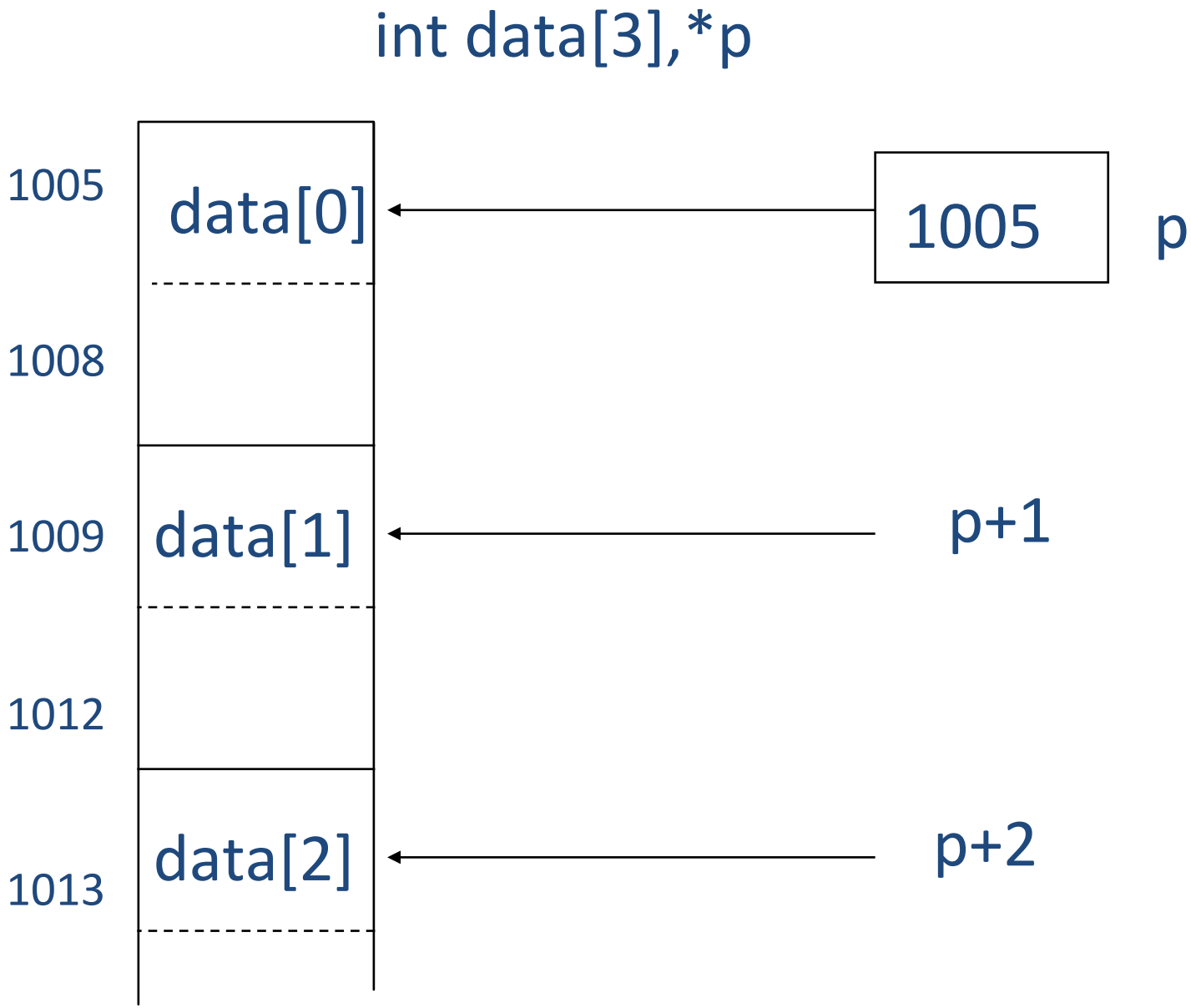
p=&data[0]; is *equivalent to* p=data;



Punteros y arrays



Punteros y arrays



Punteros y arrays

```
int i,b[20],suma;

suma=0;
for(i=0;i<20;++i)
      suma=suma+b[i];
```

```
int i,b[20],suma,*p;

suma=0;
p=b;
for(i=0;i<20;++i)
      suma=suma+*(p+i);
```

OJO:

- *(p+i)≠ *p+i
- b[i] ≠ b[0]+i

Arrays como parámetros en funciones

- Tres formas **equivalentes**
  - funcion (int a[10])
  - funcion (int a[])
  - funcion (int *a)
- Si un array se pasa como parámetro, siempre se podrá modificar dentro de la función
  - Se modifica el array, no la copia, luego los cambios persisten en la función que llama

Punteros y arrays bidimensionales

char dias[7][10]={“domingo”,”lunes”,”martes”,”miércoles”, ”jueves”,”viernes”,”sábado”}

1000	d	o	m	i	n	g	o	´\0´		
1010	l	u	n	e	s	´\0´				
1020	m	a	r	t	e	s	´\0´			
1030	m	i	e	r	c	o	l	e	s	´\0´
1040	j	u	e	v	e	s	´\0´			
1050	v	i	e	r	n	e	s	´\0´		
1060	s	a	b	a	d	o	´\0´			

- **¿Qué es dias[0]?**

Punteros y arrays bidimensionales

- dias[0] es un puntero al primer carácter de la primera fila
- cout << dias[0]; domingo

dias[0]	d	o	m	i	n	g	o	´\0´		
dias[1]	l	u	n	e	s	´\0´				
dias[2]	m	a	r	t	e	s	´\0´			
dias[3]	m	i	e	r	c	o	l	e	s	´\0´
dias[4]	j	u	e	v	e	s	´\0´			
dias[5]	v	i	e	r	n	e	s	´\0´		
dias[6]	s	a	b	a	d	o	´\0´			

Punteros y arrays bidimensionales

```
char dias[7][10]={"domingo","lunes","martes","miércoles",
        "jueves","viernes","sábado"};

char *DIAS[7]={"domingo","lunes","martes","miércoles",
        "jueves","viernes","sábado"};
```

*DIAS[7] es un array de siete punteros a cadenas de caracteres

Arrays bidimensionales y funciones

```
#include <iostream>
using namespace std;
void asignavalor(int mat[][4], int k, int l);
int main(){
	int mat[3][4],i,j;
	asignavalor(mat,3,4);
	for(i=0;i<3;++i){
		for(j=0;j<4;++j)
			cout<<mat[i][j];
		cout<<"\n";
	}
	return 0;
}
```

```
void asignavalor(int mat[][4],int k, int l){
	int i,j;
	for(i=0;i<k;++i)
		for(j=0;j<l;++j)
			mat[i][j]=j+i*l;
}
```

Arrays bidimensionales y funciones

```
#include <iostream>
using namespace std;
void asignavalor(int *mat, int k, int l);
int main(){
	int mat[3][4],i,j;
	asignavalor(&mat[0][0],3,4);
	for(i=0;i<3;++i){
		for(j=0;j<4;++j)
			cout<<mat[i][j];
		cout<<"\n";
	}
	return 0;
}
```

```
void asignavalor(int *mat,int k, int l){
	int i,j;
	for(i=0;i<k;++i)
		for(j=0;j<l;++j)
			*(mat+i*l+j)=j+i*l;
}
```

Funciones que retornan punteros

- La declaración:

int * funcion(float a)

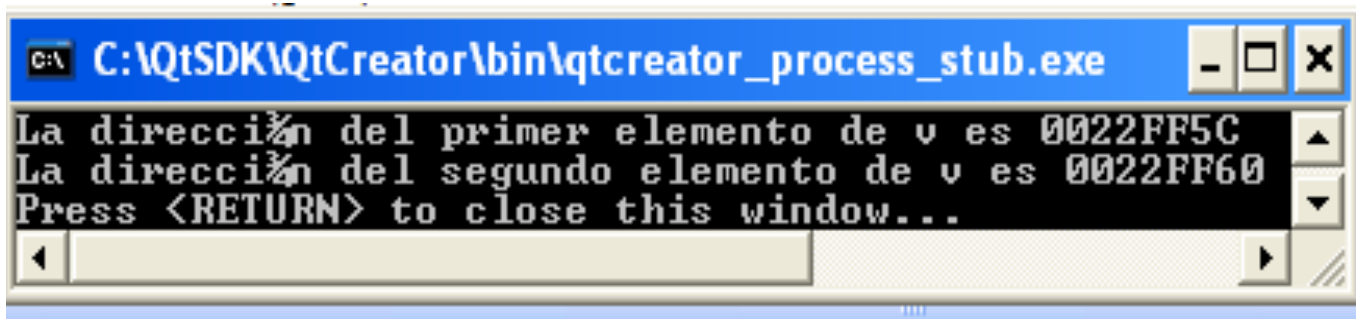
- Declara que la función tiene como parámetro un float y devuelve un puntero a un entero

Funciones que retornan punteros

```
#include <iostream>
using namespace std;
float *DevuelveDireccion(float vector[5]) ;

int main(){
	float v[5] ;
	cout<< "La dirección del primer elemento de v es "<< &v[0] << endl;
	cout<< "La dirección del primer elemento de v es " DevuelveDireccion(v) << endl;
	return 0;
	}
float *DevuelveDireccion(float vector[5])
{
	float *pun;
	pun= &vector[1]
	return (pun);
}
```

Funciones que retornan punteros

```
#include <iostream>
using namespace std;
float *DevuelveDireccion(float vector[5]) ;

int main(){
        float v[5] ;
        cout<< "La dirección del primer elemento de v es "<< &v[0] << endl;
        cout<< "La dirección del segundo elemento de v es " << DevuelveDireccion(v) << endl;
        return 0;
        }
float *DevuelveDireccion(float vector[5])
{
        float *pun;
        pun= &vector[1]
        return (pun);
}
```



INFORMÁTICA INDUSTRIAL

PROGRAMACIÓN BÁSICA C++ (III)

M. Abderrahim, A. Castro, J. C. Castillo
Departamento de Ingeniería de Sistemas y Automática

uc3m | Universidad Carlos III de Madrid