UNIVERSIDAD DE LOS LLANOS

13 Gestión de la Memoria

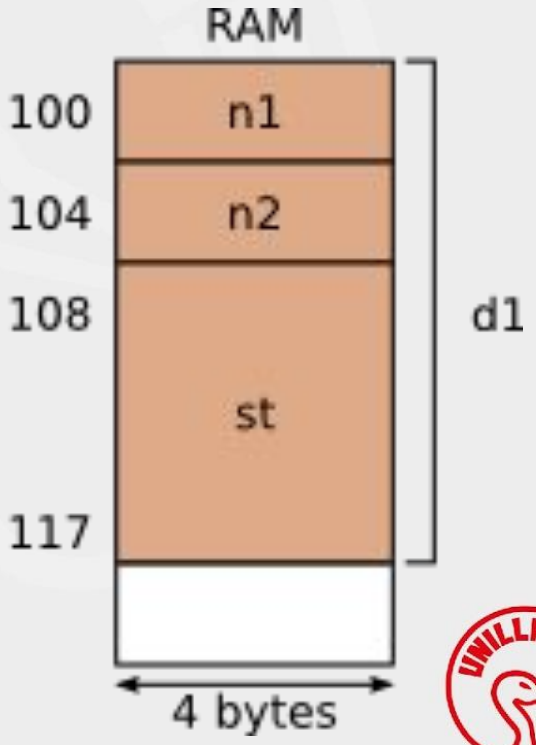
Néstor Suat-Rojas. Ing. Msc.
nestor.suat@unillanos.edu.co

Escuela de Ingeniería
Facultad de Ciencias Básicas e Ingeniería

Dirección de memoria

Todos los datos se almacenan a partir de una dirección de memoria.

```
class contact{
public:
    int n1;
    float n2;
    char st[10];
}d1;
```

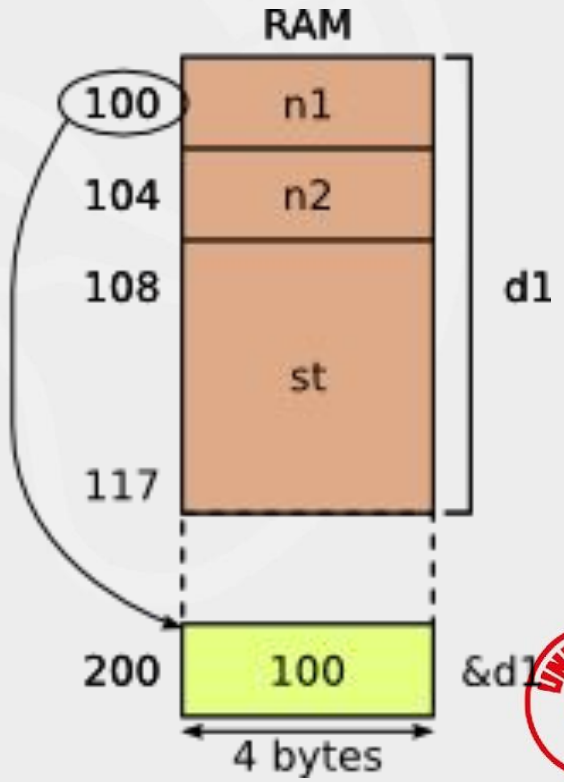


UNIVERSIDAD DE LOS LLANOS

La indirección

Acceder a los datos que apunta una variable.

```
class contact{
public:
    int n1;
    float n2;
    char st[10];
}d1;
```

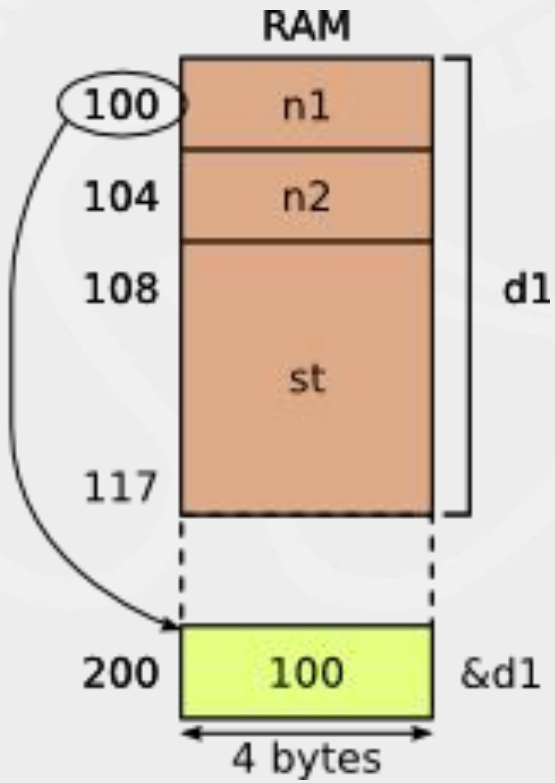


UNIVERSIDAD DE LOS LLANOS

Puntero

Los punteros son variables cuyos datos son direcciones de memoria.

```
class contact{
public:
    int n1;
    float n2;
    char st[10];
}d1;
```



UNIVERSIDAD DE LOS LLANOS

Puntero

Los punteros son variables cuyos datos son direcciones de memoria.

```
1 TIPO * nombre_puntero ;
2
3 int *p;
4 char *pchar;
```

UNIVERSIDAD DE LOS LLANOS

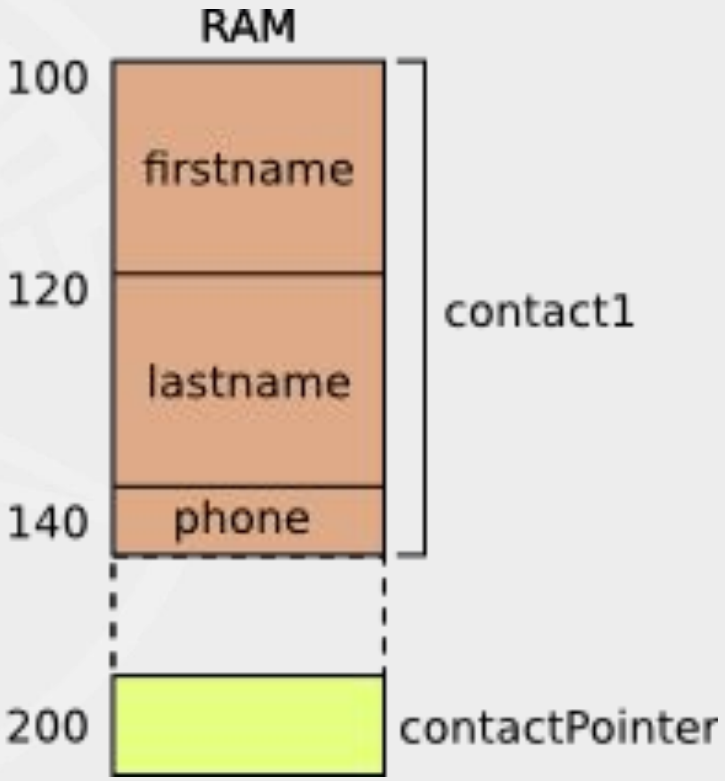
Puntero

Tipo T	Tamaño (bytes) [a]	Puntero a T	Tamaño (bytes)	Ejemplo de uso
int	4	int *	4	int *a, *b, *c;
unsigned int	4	unsigned int *	4	unsigned int *d, *e, *f;
short int	2	short int *	4	short int *g, *h, *i;
unsigned short int	2	unsigned short int *	4	unsigned short int *j, *k, *l;
long int	4	long int *	4	long int *m, *n, *o;
unsigned long int	4	unsigned long int *	4	unsigned long int *p, *q, *r;
char	1	char *	4	char *s, *t;
unsigned char	1	unsigned char *	4	unsigned char *u, *v;
float	4	float *	4	float *w, *x;
double	8	double *	4	double *y, *z;
long double	8	long double *	4	long double *a1, *a2;

[a] Valores de referencia, pueden cambiar dependiendo de la plataforma

Puntero

```
class Contact{
public:
    char firstname[20];
    char lastname[20];
    unsigned int phone;
};

Contact contact1;
Contact *contactPointer;
```



UNIVERSIDAD DE LOS LLANOS

Puntero

Asignación de una dirección a un puntero

```c
#include <stdio.h>
int main()
{
    int num1, num2;
    int *ptr1, *ptr2;

    ptr1 = &num1;
    ptr2 = &num2;

    num1 = 10;
    num2 = 20;

    ptr1 = ptr2;
    ptr2 = NULL;

    return 0;
}
```

Puntero

Asignación de una dirección a un puntero

```c
#include <stdio.h>
int main()
{
    int num1, num2;
    int *ptr1, *ptr2;

    ptr1 = &num1;
    ptr2 = &num2;

    num1 = 10;
    num2 = 20;

    ptr1 = ptr2;
    ptr2 = NULL;

    return 0;
}
```

UNIVERSIDAD DE LOS LLANOS

Puntero

Diferencias entre * y &.

- & ⟶ Obtiene la dirección en memoria

```
char a;          /* Variable 'a' de tipo char */

printf("la direccion de memoria de 'a' es: %p \n", &a);
```

- * ⟶ Obtener lo apuntado por un puntero

```
char a;          /* Variable 'a' de tipo char */
char *pchar;     /* Puntero a char 'pchar' */

pchar = &a;      /* 'pchar' <- @ de 'a' */

printf("la direccion de memoria de 'a' es: %p \n", &a);
printf("y su contenido es : %c \n", *pchar);
```

UNIVERSIDAD DE LOS LLANOS

Puntero

La indirección a través de los punteros:

- Acceder a los datos que apunta.

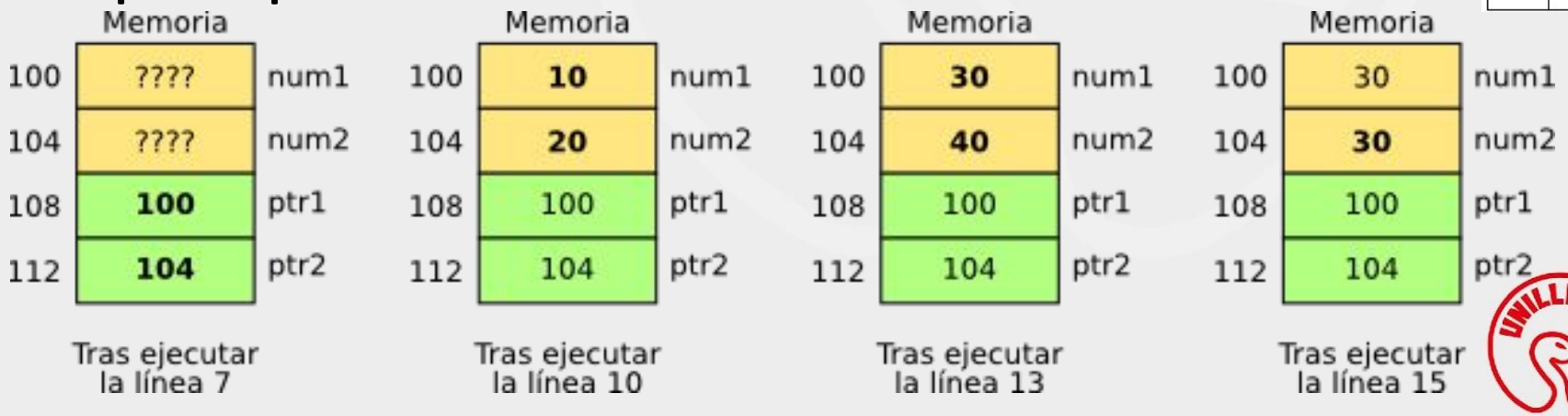
```
1   #include <stdio.h>
2   int main(int argc, char** argv)
3   {
4       int num1, num2;
5       int *ptr1, *ptr2;
6
7       ptr1 = &num1;
8       ptr2 = &num2;
9
10      num1 = 10;
11      num2 = 20;
12
13      *ptr1 = 30;
14      *ptr2 = 40;
15
16      *ptr2 = *ptr1;
17
18      return 0;
19  }
```

Puntero

La indirección a través de los punteros:

- Acceder a los datos que apunta.

```
1   #include <stdio.h>
2   int main(int argc, char** argv)
3   {
4       int num1, num2;
5       int *ptr1, *ptr2;
6
7       ptr1 = &num1;
8       ptr2 = &num2;
9
10      num1 = 10;
11      num2 = 20;
12
13      *ptr1 = 30;
14      *ptr2 = 40;
15
16      *ptr2 = *ptr1;
17
18      return 0;
19  }
```



UNIVERSIDAD DE LOS LLANOS

Puntero

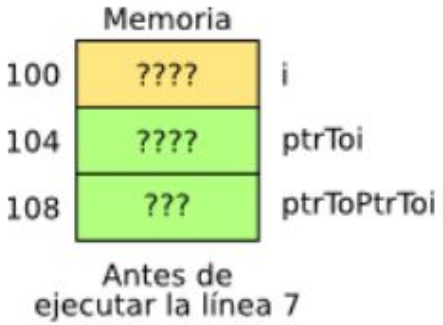
```cpp
// Definición de la clase
class Coordenadas{
public:
    float x;
    float y;
    float z;
};

int main(){
    // Declaración de dos objetos
    Coordenadas ubicacion1, ubicacion2;
    // Declaración de dos punteros
    Coordenadas *ptr1, *ptr2;

    // Asignación de direcciones a los punteros
    ptr1 = &ubicacion1;
    ptr2 = &ubicacion2;

    // Asignación de valores al primer objetos
    ptr1->x = 3.5;
    ptr1->y = 5.5;
    ptr1->z = 10.5;

    // Copia de valores al segundo objeto
    ptr2->x = ptr1->x;
    ptr2->y = ptr1->y;
    ptr2->z = ptr1->z;

    cout << ptr1->x << "\t" << ptr1->y << "\t" << ptr1->z << endl;
    cout << ptr2->x << "\t" << ptr2->y << "\t" << ptr2->z << endl;
    return 0;
}
```

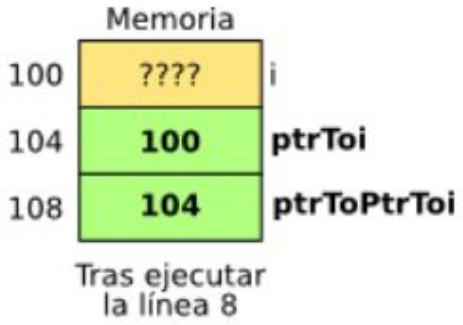
Puntero a punteros

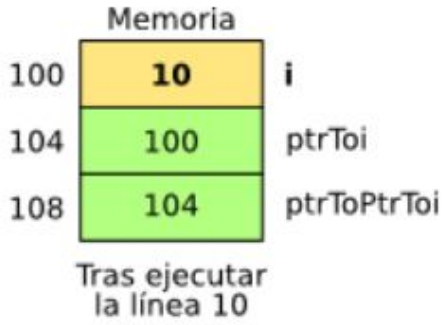
```c
#include <stdio.h>
int main()
{
    int i;
    int *ptrToi;            /* Puntero a entero */
    int **ptrToPtrToi;      /* Puntero a puntero a entero */

    ptrToPtrToi = &ptrToi; /* Puntero contiene dirección de puntero */
    ptrToi = &i;            /* Puntero contiene dirección de entero */

    i = 10;                 /* Asignación directa */
    *ptrToi = 20;           /* Asignación indirecta */
    **ptrToPtrToi = 30;     /* Asignación con doble indirección */

    return 0;
}
```

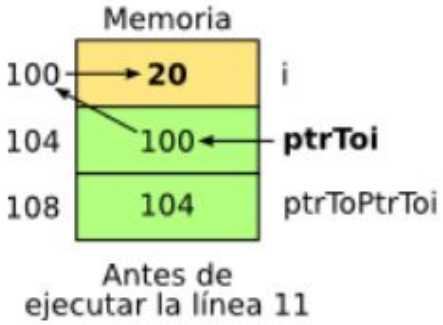
Puntero a punteros

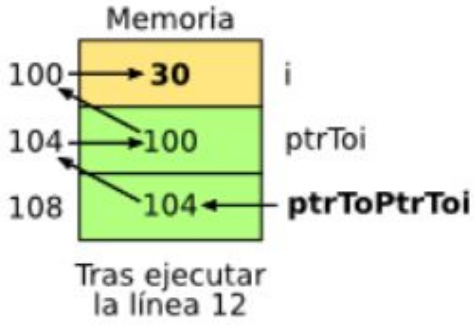
```
1   #include <stdio.h>
2   int main()
3   {
4       int i;
5       int *ptrToi;
6       int **ptrToPtrToi;
7
8       ptrToPtrToi = &ptrToi;
9       ptrToi = &i;
10
11      i = 10;
12      *ptrToi = 20;
13      **ptrToPtrToi = 30;
14
15      return 0;
16  }
```











VERSIDAD
LOS LLANOS®

Parámetro por referencia

Cuando se pasa una variable por referencia, el compilador no pasa la copia de un valor del argumento; sino que pasa una referencia que indica a la función dónde se encuentra la variable en memoria.

```
1 int a;
2 funcion(&a);
3 printf("%d", a);
```

```
5 funcion(int *x){
6     *x = 3;
7     return 0;
8 }
```

Parámetro por referencia

```
 9
10 int main() {
11     int num = 5;
12     printf("Old: %d\n", num);
13     cuadrado(&num);
14     printf("New: %d\n", num);
15     return 0;
16 }
17
18 void cuadrado(int *n){
19     int a = *n;
20     a = a*a;
21     *n = a;
22 }
```

UNIVERSIDAD DE LOS LLANOS

```cpp
// Definición de la clase
class Coordenadas{
public:
    float x;
    float y;
    float z;
};
// Definición de función que calcula la distancia entre dos puntos
float distancia(Coordenadas *a_ptr, Coordenadas *b_ptr){
    return sqrtf( x: pow( x: a_ptr->x - b_ptr->x,  y: 2.0) +
                  pow( x: a_ptr->y - b_ptr->y,  y: 2.0) +
                  pow( x: a_ptr->z - b_ptr->z,  y: 2.0));
}

int main(){
    // Declaración e inicialización de dos objetos
    Coordenadas ubicacion1 = {  .x: 3.5e-120,  .y: 2.5,  .z: 1.5};
    Coordenadas ubicacion2 = {  .x: 5.3e-120,  .y: 3.1,  .z: 6.3};
    float d; // Almacenar el resultado

    // LLamada a la función con los dos objetos
    d = distancia(&ubicacion1, &ubicacion2);
    cout << d;
    return 0;
}
```

Objetos como parámetros

UNIVERSIDAD DE LOS LLANOS

Bibliografía

C++ Programming: An Object-Oriented Approach. B. Forouzan & R. Gilberg. 2020. McGraw-Hill education.

UNIVERSIDAD DE LOS LLANOS

Gracias...

UNIVERSIDAD DE LOS LLANOS®