Fundamentos de programación

Punteros

Facultad de Ingeniería / Escuela de Informática
Universidad Andrés Bello, Viña del Mar.

Punteros

- Es un dato que contiene una dirección de memoria
- **NULL**: es una dirección especial para indicar que un puntero no apunta a ninguna dirección.

UNIVERSIDAD ANDRES BELLO

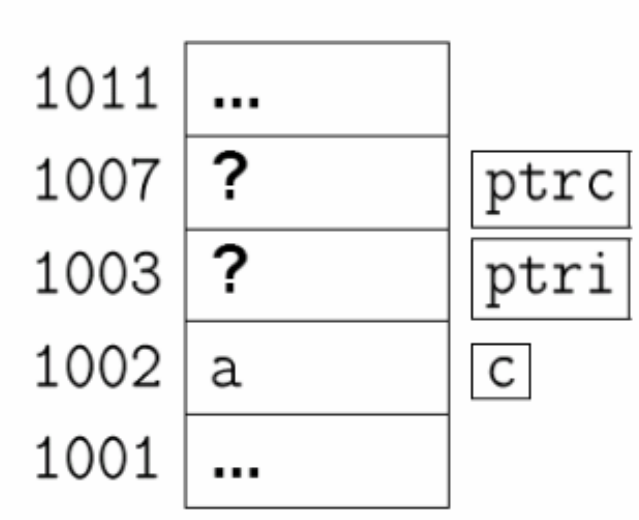
Declaración

- `tipo *nombre`
- Se reserva memoria para albergar una dirección de memoria, no para almacenar el dato al que apunta el puntero

Espacio de memoria

- el espacio usado es el mismo independientemente del tipo de dato al que apunta

```
char c = 'a';

char *ptrc;

int *ptri;
```



Operadores

Dirección

- operador &
- devuelve la dirección de memoria donde comienza la variable

```
int i;
int *ptr;
...
ptr = &i;
```

Operadores

Indirección

- operador *
- se usa para acceder a los objetos a los que apunta un puntero

```
char c;
char *ptr;
…
ptr = &c;
*ptr = ‘A’;      // Equivale a escribir: c = ‘A’
```

Operadores

Asignación

- operador =
- a un puntero se le puede asignar una dirección de memoria concreta, la dirección de una variable o el contenido de otro puntero

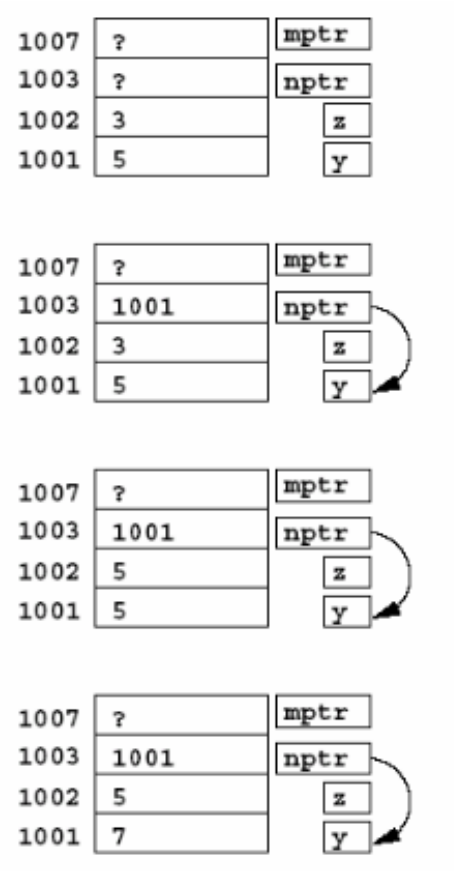
```
int *ptr;
...
ptr = 0x1F3CE00A;
...
ptr = NULL;
```

```
char c;
char *ptr;
...
ptr = &c;
```

```
char c;
char *ptr1;
char *ptr2;
…
ptr1 = &c;
ptr2 = ptr1;
```

Ejemplo

```
int main ()
{
  int y = 5;
  int z = 3;
  int *nptr;
  int *mptr;


  nptr = &y;



  z = *nptr;



  *nptr = 7;
```

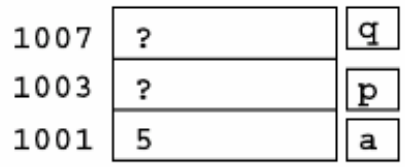


UNIVERSIDAD ANDRES BELLO

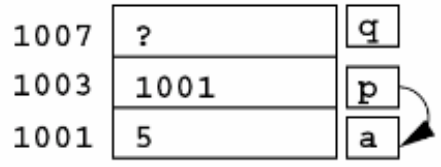
Puntero a puntero

- Es un puntero que contiene la dirección de memoria de otro puntero

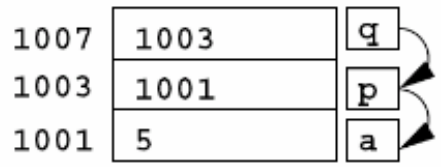
```
int main ()
{
  int a = 5;
  int *p;    // Puntero a entero
  int **q;   // Puntero a puntero
```




```
  p = &a;
```




```
  q = &p;
}
```



Ejercicio 1

Desarrolle una función en C que permita intercambiar dos números enteros usando punteros.

Ejercicio 2

Indique el valor de la variable **a** después de la ejecución de este código:

```
int a = 5;
int *p, int **q;
p = & a;
q = & p;
a = *p*2 + **q;
```

Reserva de memoria

- Al reservar puede que no quede espacio libre suficiente, por lo que se debe comprobar que no haya fallo de memoria (chequear si la dirección devuelta es distinta de NULL)
- `malloc(tam)`

Liberación de memoria

- Después de usar la memoria reservada hay que liberar el espacio, sino no se puede volver a utilizar

Ejemplo

```
#include <stdio.h>
#include <stdlib.h>

float media (float v[], int n)
{
    int   i;
    float suma = 0;

    for (i=0; i<n; i++)
        suma += v[i];

    return suma/n;
}
```

Ejemplo

```
int main()
{
    int i;
    int n;
    float *v;

    printf("Número de elementos del vector: ");
    scanf("%d",&n);

    // Creación del vector

    v = malloc(n*sizeof(float));

    // Manejo del vector

    for (i=0; i<n; i++)
        v[i] = i;

    printf("Media = %f\n", media(v,n));

    // Liberación de memoria

    free(v);

    return 0;
}
```

UNIVERSIDAD ANDRES BELLO

Ejercicio 3

Desarrolle una función en C que copie una cadena de caracteres en otra cadena (sin usar `strcpy`).

UNIVERSIDAD ANDRES BELLO

Ejercicio 4

Desarrolle un programa en C que lea una frase y escriba cada una de las palabras de su contenido al revés.

Ejercicio 5

Se pide crear un programa que pida una serie de números al usuario y halle el máximo, el mínimo y el promedio de todos los datos. Para ello se debe crear una variable puntero tipo `float`, pedir al usuario que introduzca el número de datos, y después los datos a calcular. Se debe reservar memoria de forma dinámica para almacenar el vector de datos.