Sesión 5

Aprende C desde cero

- Datos en memoria
  - Punteros
  - Parámetros por Referencia
- Procedimientos

Datos en memoria

Punteros

Los datos se almacena en algún tipo de memoria: registros, caché, memoria principal…

Para acceder a esos datos utilizamos variables, que son alias de zonas en la memoria

C ofrece al programador acceder a estas zonas mediante el uso de punteros para permitir el manejo de grandes cantidades de datos

Concretamente, para indicar a una función a que zonas de memoria tiene acceso:

- Enviar vectores o cadenas a funciones
- Permitir que las variables sean de entrada/salida

Punteros

Los datos se almacena en algún tipo de memoria: registros, caché, memoria principal…

Para acceder a esos datos utilizamos variables, que son alias de zonas en la memoria

C ofrece al programador acceder a estas zonas mediante el uso de punteros para permitir el manejo de grandes cantidades de datos

Concretamente, para indicar a una función a que zonas de memoria tiene acceso:

- Enviar vectores o cadenas a funciones
- Permitir que las variables sean de entrada/salida

Punteros

Los datos se almacena en algún tipo de memoria: registros, caché, memoria principal…

Para acceder a esos datos utilizamos variables, que son alias de zonas en la memoria

C ofrece al programador acceder a estas zonas mediante el uso de punteros para permitir el manejo de grandes cantidades de datos

Concretamente, para indicar a una función a que zonas de memoria tiene acceso:

- Enviar vectores o cadenas a funciones
- Permitir que las variables sean de entrada/salida

Punteros

Los datos se almacena en algún tipo de memoria: registros, caché, memoria principal…

Para acceder a esos datos utilizamos variables, que son alias de zonas en la memoria

C ofrece al programador acceder a estas zonas mediante el uso de punteros para permitir el manejo de grandes cantidades de datos

Concretamente, para indicar a una función a que zonas de memoria tiene acceso:

- Enviar vectores o cadenas a funciones
- Permitir que las variables sean de entrada/salida

Punteros

```
int max(int a, int b, int c, int d, int e){
     int maximo = a;
     if (b > maximo) maximo = b;
     if (c > maximo) maximo = c;
     if (d > maximo) maximo = d;
     if (e > maximo) maximo = e;
     return maximo;
}

int max2(int * v, int N){     //El * indica que el argumento es un puntero
     int maximo = v[0];
     int i;
     for( i = 1; i < N; i++){
          if(v[i] > maximo) maximo = v[i];
     }
     return maximo;
}

int main(void)
{
     int vector[5] = {3,5,2,3,1};
     int valor_maximo = max(vector[0], vector[1],vector[2],vector[3], vector[4]);
     int valor_maximo2 = max2(vector,5);
     return 0;
}
```

Punteros

Declaración:

```
tipo_dato nombre[tamaño];
tipo_dato * nombre;
```

Marca de formato (para el printf):

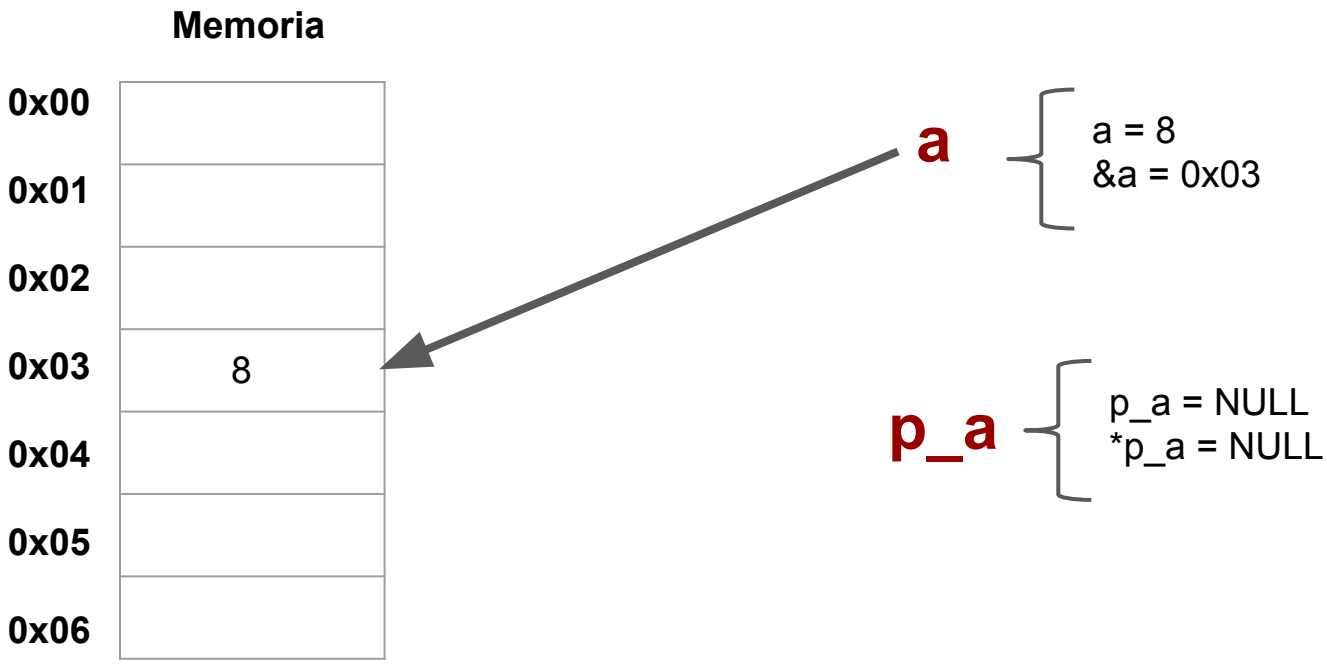
%p

```
int main(void)
{
    int a = 8;
    int *p_a;

    printf("El valor de a es %d y su dirección  %p", a, &a);
    printf("El valor de p_a es  %p",p_a);

    p_a = &a; //a p_a le damos el valor de la dirección de memoria de a

    //El valor de a no cambia
    printf("El valor de a es %d y su dirección  %p", a, &a);
    //El valor de p_a ahora es distinto
    printf("El valor de p_a es  %p y su valor %d",p_a,  *p_a);
    return 0;
}
```

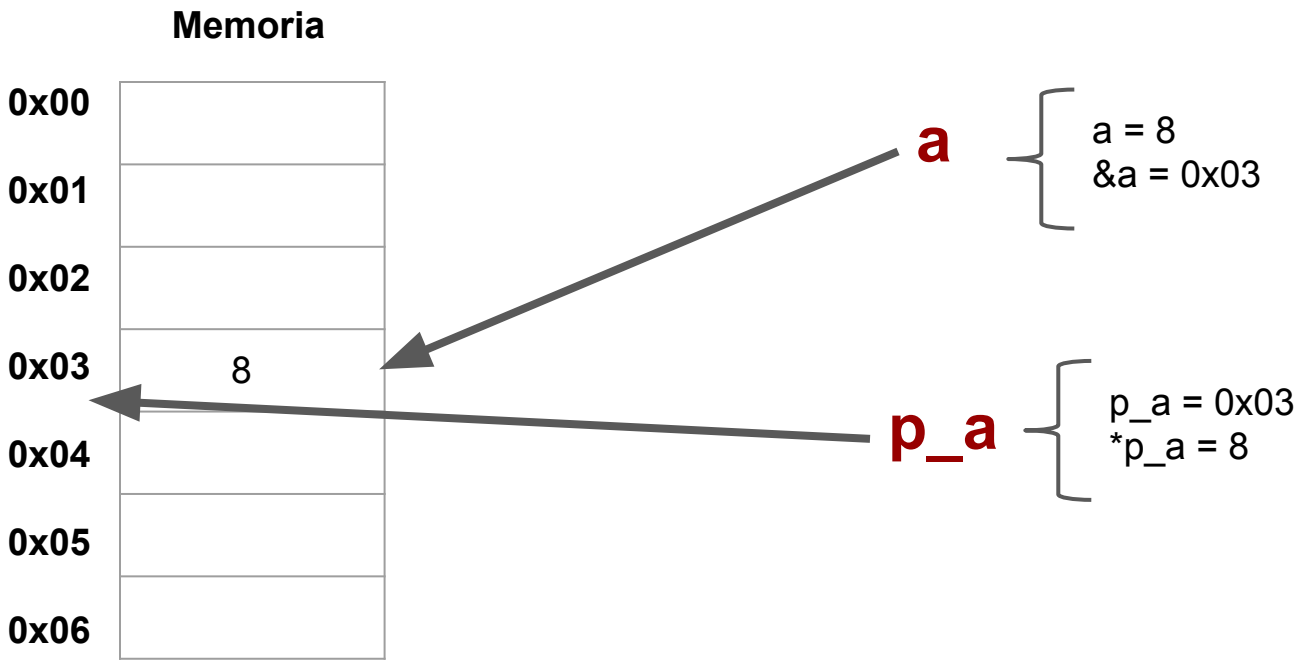
```
int a = 8;
```

```
int a = 8;
```

```
int *p_a;
```


Memoria
0x00
0x01
0x02
0x03
0x04
0x05
0x06
8
a
a = 8
&a = 0x03
p_a
p_a = NULL
*p_a = NULL

```
p_a = &a;
```


Memoria
0x00
0x01
0x02
0x03
0x04
0x05
0x06
8
a
a = 8
&a = 0x03
p_a
p_a = 0x03
*p_a = 8

Parámetros por Referencia

Parámetros por Referencia

Indica a una función que el datos de entrada es también de salida:

```
tipo_dato nombre_función (tipo_arg1 var, tipo_arg2 * var2);
```

```
int minmax(int * vec, int * min,
                  int * max, int N){

    if ( N < 1 ) return 0;
    int i = 0;

    *min = vec[0];
    *max = vec[0];
    for(i = 1; i < N; i++){
        if (vec[i] < *min){
            *min = vec[i];
        }
        if (vec[i] > *max){
            *max = vec[i];
        }
    }
    return 1;
}
```

```
int main(void)
{
    int minimo=99, maximo=-33;
    int vector[5];

    for(i = 0; i < 5; i++)
    {
        vector[i] = i;
    }

    if(minmax(vector,&minimo,&maximo,5)){
        printf("El máximo es %d y el mínimo
                es %d", maximo, minimo);
    }
    else{
        printf("El vector debe contener
                    algún elemento\n");
    }
    return 0;
}
```

Procedimientos

Procedimientos

Un procedimiento es una función que no devuelve ningún dato con el return

```
void nombre_funcion(parámetros);
```

Donde:

- **`void`**: indica que no habrá valor de respuesta.
- **`nombre_funcion`**: indica el nombre de la función. Este nombre se utilizará para invocarla.
- **`parámetros`**: es una lista de variables (argumentos) separadas por comas. Serán los valores de entrada y salida.

Ejemplo:

- Esta podría ser la definición de un procedimiento que recibe dos argumentos (num1 y num2) de tipo entero como entrada y devuelve su suma en la variable de salida “resultado”.

```
void suma(int num1, int num2, &resultado);
```

Ejercicio

Ejercicio: Calculadora 5.0

Implementa la última versión de la calculadora. En este caso debe poder operar con vectores, por lo tanto habrá que añadir el modo vector a los ya existentes.
En este modo, dados dos vectores, se podrá efectuar la suma, resta, multiplicación y división de sus elementos.

Además, las funciones implementadas en la sesión 4 (modo “números”), deberán convertirse en procedimientos. En lugar de devolver su resultado con el return, se deberá pasar una variable por referencia para almacenarlo.

Por lo tanto, con los datos de entrada siguientes: int v1[4] = {4, 8, 20, 3} y int v2[4] = {1, 6, 5, 7}, si elegimos la función resta, el resultado debería ser el siguiente:

```
Bienvenido/a a “Calcooladora v5.0”.

Modo “vector”:
    Operandos: {4 8 20 3} y {1 6 5 7}
    Operación: Resta

Resultado:   {3 2 15 -4}

¡Adiós!
```

Recuerda que puedes:

Interactuar con otros alumnos en el foro del curso:
http://mooc.uji.es/mod/forum/view.php?id=1654

Acceder a todos los materiales en el repositorio:
https://siserte@bitbucket.org/siserte/repositoriocursoc.git

Contactar con los profesores:

- adcastel@uji.es
- siserte@uji.es