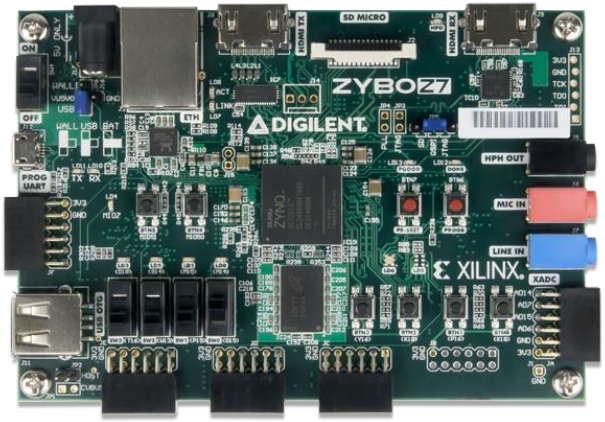
Igor Peralta
Ayudante
Departamento de Ingeniería Eléctrica
Pontificia Universidad Católica de Chile

IEE 2463
Sistemas Electrónicos Programables



Arreglos y punteros

Definición
Arreglos o Arrays

- Son colecciones indexadas de variables
- Para declarar un arreglo hay que definir:
  - Tipo de elementos
  - Nombre
  - Tamaño -> El compilador de C asigna un espacio en memoria

tipo nombre[tamaño];

- Los strings son arreglos de caracteres finalizado en el carácter nulo ‘\0’
  - char nombre_string[tamaño];

Inicialización
Arreglos o Arrays

- Formas de inicializar arrays:
  - char nombre[] = “hola”;
  - char nombre[5] = “hola”;
  - char nombre[50] = “hola”;
  - char nombre[] = {‘h’, ‘o’, ‘l’, ‘a’, ‘\0’};
  - char nombre[5] = {‘h’, ‘o’, ‘l’, ‘a’, ‘\0’};
- Notar que si bien los strings terminan en el carácter nulo. Un arreglo de ints no lo necesita:
  - int nombre[3] = {1, 2, 3};
  - int nombre[] = {1, 2, 3};
- Podemos hacer arreglos multidimensionales (arreglos de arreglos):
  - int nombre[2][3] = {{00, 01, 02}, {10, 12, 12}};

Consideraciones
Arreglos o Arrays

- LOS ARREGLOS NO PUEDEN SER ASIGNADOS CON EL OPERADOR = UNA VEZ QUE HAN SIDO DECLARADOS
  - Si queremos modificarlos habrá que modificar cada elemento individualmente
- EL PRIMER ELEMENTO DE UN ARREGLO ES EL ÍNDICE 0
- Para el manejo de strings, podemos utilizar funciones de la librería string.h

Ejemplo
Arreglos o Arrays

```c
#include <stdio.h>

int main()
{
    int arreglo_numeros[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int i;

    for(i = 0; i < 10; ++i) {
        printf("%d ", arreglo_numeros[i]);
    }
    printf("\n\n");

    arreglo_numeros[2] = 20;

    for(i = 0; i < 10; ++i) {
        printf("%d ", arreglo_numeros[i]);
    }
    printf("\n\n");

    for(i = 0; i < 10; ++i) {
        arreglo_numeros[i] = i*100;
        printf("%d ", arreglo_numeros[i]);
    }

    return 0;
}
```

Definición
Punteros

- Son variables cuyo contenido es la dirección de otra variable, arreglo o string.
- Se utiliza el operador & (ampersand) para indicar dirección
- Se utiliza el operador * como contrario al &, para indicar el contenido de la dirección. También sirve para declarar punteros.

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

Se declara un puntero llamado pc, el cual apunta a un int

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

Se declara un int llamado c y se le asigna el valor 22

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

NOTAR que se introduce como puntero

Se imprime la dirección de c

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

Se imprime el valor de c, de la misma forma que siempre lo hemos hecho con %d

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c; Se asigna al puntero pc, la dirección de c
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

El puntero corresponde a una dirección (dirección de c), mientras que su contenido, es el contenido de c

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11; Se cambia el valor de c
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

La dirección del puntero no cambia, pero sí cambia su contenido, ya que cambió c

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2; Se cambia el valor del contenido del puntero
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

https://www.programiz.com/c-programming/c-pointers

Ejercicio simple
Punteros

```c
#include <stdio.h>
int main()
{
    int* pc;
    int c;
    c = 22;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    pc = &c;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    c = 11;
    printf("Dirección de puntero pc: %p\n", pc);
    printf("Contenido de puntero pc: %d\n\n", *pc);
    *pc = 2;
    printf("Dirección de c: %p\n", &c);
    printf("Valor de c: %d\n\n", c);
    return 0;
}
```

La dirección de c no cambia, pero sí cambia su contenido

https://www.programiz.com/c-programming/c-pointers

Aplicación
Punteros

```
void swap(int x, int y)
 {
     int temp;
     temp = x;
     x = y;
     y = temp;
 }
```

¿Funciona?

Aplicación
Punteros

```
void swap(int x, int y)
 {
     int temp;
     temp = x;
     x = y;
     y = temp;
 }
```

¿Funciona?
No, porque x e y son variables locales a la función swap

Aplicación
Punteros

```
void swap(int x, int y)
 {
     int temp;
     temp = x;
     x = y;
     y = temp;
 }
```

```
void swap(int *px, int *py)
 {
     int temp;
     temp = *px;
     *px = *py;
     *py = temp;
 }
```

¿Funciona?

Aplicación
Punteros

```
void swap(int x, int y)
 {
     int temp;
     temp = x;
     x = y;
     y = temp;
 }
```

```
void swap(int *px, int *py)
 {
     int temp;
     temp = *px;
     *px = *py;
     *py = temp;
 }
```

¿Funciona?
Sí, estamos trabajando con los contenidos de direcciones en memoria

Aplicación
Punteros

¿Pudimos haber ocupado una variable global?

Aplicación
Punteros

¿Pudimos haber ocupado una variable global?

```c
#include <stdio.h>

int a;
int b;

void swap(){
    int temp = a;
    a = b;
    b = temp;
}

int main()
{
    a = 10;
    b = 20;
    swap();
    printf("a: %d  b: %d", a, b);

    return 0;
}
```

Aplicación
Punteros en funciones

¿Hay alguna función que ya hemos usado harto y utilizaba punteros?

Aplicación
Punteros en funciones

¿Hay alguna función que ya hemos usado harto y utilizaba punteros?

```c
#include <stdio.h>

int main()
{
    int valor;
    scanf("%d", &valor);
    printf("Valor: %d", valor);
}
```

Aplicación
Punteros en arrays

Si tenemos un array llamado a, el cual tiene N elementos, donde i < N, se cumplirá:

- &a[i] = a+i
- a[i] = *(a+i)

Aplicación
Punteros en arrays

Si tenemos un array llamado a, el cual tiene N elementos, donde i < N, se cumplirá:

- &a[i] = a+i
- a[i] = *(a+i)

```c
#include <stdio.h>
int main() {

  int i, x[6], sum = 0;

  printf("Enter 6 numbers: ");

  for(i = 0; i < 6; ++i) {
  // Equivalent to scanf("%d", &x[i]);
      scanf("%d", x+i);

  // Equivalent to sum += x[i]
      sum += *(x+i);
  }

  printf("Sum = %d", sum);

  return 0;
}
```