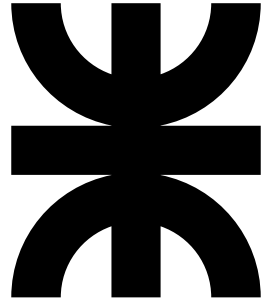
Informática II
Repaso de estructuras (`struct`)

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2019 –

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
```

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
```

- Palabra reservada `struct` define la estructura

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
```

- Palabra reservada `struct` define la estructura
- El identificador `horario` es el *rótulo* de la estructura

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
```

- Palabra reservada `struct` define la estructura
- El identificador `horario` es el *rótulo* de la estructura
- Se declara un nuevo tipo de datos (no reserva espacio en memoria) $\Longrightarrow$ `struct horario`

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};

struct horario mediodia;
```

- Palabra reservada `struct` define la estructura
- El identificador `horario` es el *rótulo* de la estructura
- Se declara un nuevo tipo de datos (no reserva espacio en memoria) $\Longrightarrow$ `struct horario`

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};

struct horario mediodia;
```

- Palabra reservada `struct` define la estructura
- El identificador `horario` es el *rótulo* de la estructura
- Se declara un nuevo tipo de datos (no reserva espacio en memoria) $\Longrightarrow$ `struct horario`
- Las variables dentro de las llaves son *miembros* de la estructura

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};

struct horario mediodia;
```

- Palabra reservada `struct` define la estructura
- El identificador `horario` es el *rótulo* de la estructura
- Se declara un nuevo tipo de datos (no reserva espacio en memoria) $\Longrightarrow$ `struct horario`
- Las variables dentro de las llaves son *miembros* de la estructura
- Miembros de una estructura: variables de tipos básicos o derivados

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};

struct horario mediodia;
```

- Palabra reservada `struct` define la estructura
- El identificador `horario` es el *rótulo* de la estructura
- Se declara un nuevo tipo de datos (no reserva espacio en memoria) $\Longrightarrow$ `struct horario`
- Las variables dentro de las llaves son *miembros* de la estructura
- Miembros de una estructura: variables de tipos básicos o derivados

Definición de estructuras (`struct`)

- Son colecciones de datos de variables relacionadas, todas bajo un mismo nombre, que pueden ser de diferentes tipos
- Son tipos de datos derivados construidos con objetos de otros tipos

Definición:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};

struct horario mediodia;
```

- Palabra reservada `struct` define la estructura
- El identificador `horario` es el *rótulo* de la estructura
- Se declara un nuevo tipo de datos (no reserva espacio en memoria) $\Longrightarrow$ `struct horario`
- Las variables dentro de las llaves son *miembros* de la estructura
- Miembros de una estructura: variables de tipos básicos o derivados

Con estructuras y punteros (estructuras auto-referenciadas) se puede generar *estructuras dinámicas de datos* tales como: listas enlazadas, pilas, colas, árboles, etc.

Ejemplos

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
struct horario hinicio, hfin;
```

Ejemplos

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
struct horario hinicio, hfin;
```

```
struct horario {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;
```

Ejemplos

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
struct horario hinicio, hfin;
```

```
struct horario {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;
```

El *rótulo* es opcional, si se omite se pueden declarar variables de estructura solo dentro de la definición.

```
struct {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;
```

Ejemplos

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
struct horario hinicio, hfin;
```

```
struct horario {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;
```

El *rótulo* es opcional, si se omite se pueden declarar variables de estructura solo dentro de la definición.

```
struct {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;
```

¿Qué otros ejemplos de estructuras se les ocurre?

Inicialización y acceso a miembros

```
struct paciente {
  char apellido[20];
  char nombre[20];
  int edad;
  float peso;
  float altura;
};
```

Inicialización y acceso a miembros

```
struct paciente {
  char apellido[20];
  char nombre[20];
  int edad;
  float peso;
  float altura;
};
```

¿Cómo se inicializa una estructura? (inicialización de entero: `int a = 2;`)

Inicialización y acceso a miembros

```
struct paciente {
  char apellido[20];
  char nombre[20];
  int edad;
  float peso;
  float altura;
};
```

¿Cómo se inicializa una estructura? (inicialización de entero: `int a = 2;`)

```
/* Inicialización de la estrucura */
struct paciente jperez = { "Perez", "Juan",
  48, 88.5, 1.85 };
```

Inicialización y acceso a miembros

```
struct paciente {
  char apellido[20];
  char nombre[20];
  int edad;
  float peso;
  float altura;
};
```

¿Cómo se inicializa una estructura? (inicialización de entero: `int a = 2;`)

```
/* Inicialización de la estrucura */
struct paciente jperez = { "Perez", "Juan",
  48, 88.5, 1.85 };

/* Asignación de miembros */
struct paciente cdiaz;
cdiaz.edad = 39;
strcpy(rdiaz.apellido, "Díaz");
```

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (`&`) de una variable de estructura

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (`&`) de una variable de estructura
- Utilizar el operador `sizeof` para determinar el tamaño del tipo de datos estructura

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (`&`) de una variable de estructura
- Utilizar el operador `sizeof` para determinar el tamaño del tipo de datos estructura

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (`&`) de una variable de estructura
- Utilizar el operador `sizeof` para determinar el tamaño del tipo de datos estructura
- Acceder a los miembros de una estructura

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (`&`) de una variable de estructura
- Utilizar el operador `sizeof` para determinar el tamaño del tipo de datos estructura
- Acceder a los miembros de una estructura

Ejemplos:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;
```

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (&) de una variable de estructura
- Utilizar el operador `sizeof` para determinar el tamaño del tipo de datos estructura
- Acceder a los miembros de una estructura

Ejemplos:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;

hinicio.hora = 13;
hinicio.minuto = 0;
hinicio.segundo = 0;
```

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (&) de una variable de estructura
- Utilizar el operador `sizeof` para determinar el tamaño del tipo de datos estructura
- Acceder a los miembros de una estructura

Ejemplos:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;

hinicio.hora = 13;
hinicio.minuto = 0;
hinicio.segundo = 0;

struct hora h = hinicio;
```

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (`&`) de una variable de estructura
- Utilizar el operador `sizeof` para determinar el tamaño del tipo de datos estructura
- Acceder a los miembros de una estructura

Ejemplos:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;

hinicio.hora = 13;
hinicio.minuto = 0;
hinicio.segundo = 0;

struct hora h = hinicio;
struct hora *hptr = &hinicio;
```

Operaciones válidas

- Asignar variables de estructuras a otras de mismo tipo
- Tomar la dirección (&) de una variable de estructura
- Utilizar el operador `sizeof` para determinar el tamaño del tipo de datos estructura
- Acceder a los miembros de una estructura

Ejemplos:

```
struct horario {
  int horas;
  int minutos;
  int segundos;
} hinicio, hfin;

hinicio.hora = 13;
hinicio.minuto = 0;
hinicio.segundo = 0;

struct hora h = hinicio;
struct hora *hptr = &hinicio;
size_t hstruct_size = sizeof(struct horario); /* sizeof(hfin) */
```

`typedef` con estructuras

La palabra reservada `typedef` permite crear alias para tipos de datos definidos anteriormente

`typedef` con estructuras

La palabra reservada `typedef` permite crear alias para tipos de datos definidos anteriormente

Ejemplos:

```
typedef long unsigned int size_t;
```

`typedef` con estructuras

La palabra reservada `typedef` permite crear alias para tipos de datos definidos anteriormente

Ejemplos:

```
typedef long unsigned int size_t;
```

`typedef` en definición de estructuras

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
typedef struct horario horario_t;
```

`typedef` con estructuras

La palabra reservada `typedef` permite crear alias para tipos de datos definidos anteriormente

Ejemplos:

```
typedef long unsigned int size_t;
```

`typedef` en definición de estructuras

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
typedef struct horario horario_t;

horario_t hinicio, hfin;
```

`typedef` con estructuras

La palabra reservada `typedef` permite crear alias para tipos de datos definidos anteriormente

Ejemplos:

```
typedef long unsigned int size_t;
```

`typedef` en definición de estructuras

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
typedef struct horario horario_t;

horario_t hinicio, hfin;
```

```
typedef struct {
  int horas;
  int minutos;
  int segundos;
} horario_t;
```

`typedef` con estructuras

La palabra reservada `typedef` permite crear alias para tipos de datos definidos anteriormente

Ejemplos:

```
typedef long unsigned int size_t;
```

`typedef` en definición de estructuras

```
struct horario {
  int horas;
  int minutos;
  int segundos;
};
typedef struct horario horario_t;

horario_t hinicio, hfin;
```

```
typedef struct {
  int horas;
  int minutos;
  int segundos;
} horario_t;

horario_t hinicio, hfin;
```

Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];
```

Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];

turno_tarde[2].peso = 78.5;
```

Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];

turno_tarde[2].peso = 78.5;
```

Puntero a estructuras

```
struct paciente *jperez;

(*jperez).peso = 78.5;
```

Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];

turno_tarde[2].peso = 78.5;
```

Puntero a estructuras

```
struct paciente *jperez;

(*jperez).peso = 78.5;
jperez->altura = 1.76;
```

Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];

turno_tarde[2].peso = 78.5;
```

Puntero a estructuras

```
struct paciente *jperez;

(*jperez).peso = 78.5;
jperez->altura = 1.76;
```

Operador flecha ->

Anidamiento de estructuras

```
typedef struct {
  int dia, mes, anio;
} fecha_t;
```

Anidamiento de estructuras

```
typedef struct {
  int dia, mes, anio;
} fecha_t;

typedef {
  char apellido[20];
  char nombre[20];
  int edad;
  float peso;
  float altura;
  fecha_t nacimiento;
} paciente_t;
```

Anidamiento de estructuras

```
typedef struct {
  int dia, mes, anio;
} fecha_t;

typedef {
  char apellido[20];
  char nombre[20];
  int edad;
  float peso;
  float altura;
  fecha_t nacimiento;
} paciente_t;



paciente_t jperez;
```

Anidamiento de estructuras

```
typedef struct {
  int dia, mes, anio;
} fecha_t;

typedef {
  char apellido[20];
  char nombre[20];
  int edad;
  float peso;
  float altura;
  fecha_t nacimiento;
} paciente_t;
```

```
paciente_t jperez;
jperez.nacimiento.anio = 1988;
```

Pasaje a funciones

- Las estructuras se pasan por valor a las funciones

Pasaje a funciones

- Las estructuras se pasan por valor a las funciones
- Se puede pasar un puntero de estructura a una función simulando llamada por referencia

Pasaje a funciones

- Las estructuras se pasan por valor a las funciones
- Se puede pasar un puntero de estructura a una función simulando llamada por referencia
- Una forma (no ortodoxa) de pasar un arreglo a una función por valor es envolverlo en una estructura