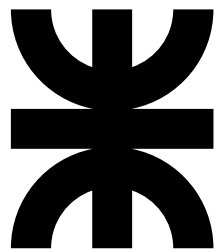
Informática II

Estructuras

Gonzalo F. Pérez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2018 –

Estructuras

Definición

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

Estructuras

Definición

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

Las *estructuras* son tipos de datos derivados, construidas con objetos de otros tipos

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

Estructuras

Definición

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

Las *estructuras* son tipos de datos derivados, construidas con objetos de otros tipos

- Palabra reservada **`struct`** define la estructura

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

Estructuras

Definición

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

Las *estructuras* son tipos de datos derivados, construidas con objetos de otros tipos

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

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

Estructuras

Definición

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

Las *estructuras* son tipos de datos derivados, construidas con objetos de otros tipos

```
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)

Estructuras

Definición

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

Las *estructuras* son tipos de datos derivados, construidas con objetos de otros tipos

```
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)
- Las variables dentro de las llaves son *miembros* de la estructura

Estructuras

Definición

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

Las *estructuras* son tipos de datos derivados, construidas con objetos de otros tipos

```
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)
- Las variables dentro de las llaves son *miembros* de la estructura
- Miembros de una estructura: variables de tipos básicos o derivados

Estructuras

Definición

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

Las *estructuras* son tipos de datos derivados, construidas con objetos de otros tipos

```
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)
- 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 de datos* dinámicas tales como: listas enlazadas, pilas, colas, árboles, etc.

Estructuras

Definición – Ejemplos

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

Estructuras

Definición – Ejemplos

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

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

Estructuras
Definición – 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 puede declarar variables de estructura solo dentro de la definición.

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

Estructuras
Definición – 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 puede declarar variables de estructura solo dentro de la definición.

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

Ver más ejemplos.

Estructuras

Inicialización y acceso a miembros

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

Estructuras

Inicialización y acceso a miembros

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

```
. . .
struct paciente jperez = { "Perez", "Juan",
  48, 1.85, 88.5 };
. . .
struct paciente cdiaz;
cdiaz.edad = 39;
  strcpy(rdiaz.apellido, "Díaz");
```

Estructuras

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

Estructuras

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;
```

Estructuras

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;
```

Estructuras

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;
```

Estructuras

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;
```

Estructuras
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);
```

Estructuras

`typedef` con estructuras

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

Estructuras

`typedef` con estructuras

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

Ejemplos:

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

Estructuras

`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;
```

Estructuras

`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;
```

Estructuras

`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;
```

Estructuras

`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;
```

Estructuras
Arreglos y punteros

Arreglo de estructuras

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

Estructuras
Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];
. . .
turno_tarde[2].peso = 78.5;
```

Estructuras
Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];
. . .
turno_tarde[2].peso = 78.5;
```

Puntero a estructuras

```
struct paciente *jperez;
. . .
(*jperez).nombre = "Juan";
```

Estructuras
Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];
. . .
turno_tarde[2].peso = 78.5;
```

Puntero a estructuras

```
struct paciente *jperez;
. . .
(*jperez).nombre = "Juan";
jperez->apellido = "Perez";
```

Estructuras
Arreglos y punteros

Arreglo de estructuras

```
struct paciente turno_tarde[10];
. . .
turno_tarde[2].peso = 78.5;
```

Puntero a estructuras

```
struct paciente *jperez;
. . .
(*jperez).nombre = "Juan";
jperez->apellido = "Perez";
```

Operador flecha `->`

Estructuras

Anidamiento de estructuras

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

Estructuras
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;
```

Estructuras

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;
```

Estructuras
Pasaje a funciones

- Las estructuras se pasan por valor a las funciones

Estructuras
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

Estructuras
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