Informática I

Estructuras - 1ra Parte

Claudio J. Paz

24 de septiembre de 2018

Estructuras

Definición

Estructuras

Definición

```
struct punto2D {
  float x;
  float y;
};
```

Estructuras

Definición

```
struct punto2D {
  float x;
  float y;
};
```

- `struct` es la palabra clave para indicarle al compilador que se definirá una estructura

Estructuras

Definición

```
struct punto2D {
  float x;
  float y;
};
```

- `struct` es la palabra clave para indicarle al compilador que se definirá una estructura
- `punto2D` es la etiqueta de la estructura

```
struct punto2D {
  float x;
  float y;
};
```

- `struct` es la palabra clave para indicarle al compilador que se definirá una estructura
- `punto2D` es la etiqueta de la estructura
- `x` e `y` son los miembros de la estructura. Tiene que definirse su tipo. Pueden haber todos los miembros que se quieran

Estructuras

Definición, Declaración y Uso

Estructuras

Definición, Declaración y Uso

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

int main(void)
{

  struct punto2D punto1;


  punto1.x = 2;
  punto1.y = 3;



  printf("El punto tiene coordenadas (%.1f, %.1f)\n", punto1.x, punto1.y);

  return 0;
}
```

Estructuras

Definición, Declaración y Uso

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

int main(void)
{

  struct punto2D punto1;


  punto1.x = 2;
  punto1.y = 3;



  printf("El punto tiene coordenadas (%.1f, %.1f)\n", punto1.x, punto1.y);

  return 0;
}
```

```
El punto tiene coordenadas (2.0, 3.0)
```

Estructuras

Definición, Declaración y Uso

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

int main(void)
{

  struct punto2D punto1;
  struct punto2D punto2;

  punto1.x = 2;
  punto1.y = 3;

  punto2 = punto1;

  printf("El punto tiene coordenadas (%.1f, %.1f)\n", punto2.x, punto2.y);

  return 0;
}
```

Estructuras

Definición, Declaración y Uso

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

int main(void)
{

  struct punto2D punto1;
  struct punto2D punto2;

  punto1.x = 2;
  punto1.y = 3;

  punto2 = punto1;

  printf("El punto tiene coordenadas (%.1f, %.1f)\n", punto2.x, punto2.y);

  return 0;
}
```

```
El punto tiene coordenadas (2.0, 3.0)
```

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

struct punto3D {
  float x;
  float y;
  float z;
};

int main(void)
{

  struct punto2D punto1;
  struct punto3D punto2;

  punto1.x = 2;
  punto1.y = 3;

  punto2 = punto1;

  printf("El punto tiene coordenadas (%.1f, %.1f)\n", punto2.x, punto2.y);

  return 0;
}
```

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

struct punto3D {
  float x;
  float y;
  float z;
};

int main(void)
{

  struct punto2D punto1;
  struct punto3D punto2;

  punto1.x = 2;
  punto1.y = 3;

  punto2 = punto1;

  printf("El punto tiene coordenadas (%.1f, %.1f)\n", punto2.x, punto2.y);

  return 0;
}
```

```
struct-03.c: In function 'main':
struct-03.c:23:10: error: incompatible types when assigning to type 'struct
     punto3D' from type 'struct punto2D'
   punto2 = punto1;
          ^
```

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

struct punto3D {
  float x;
  float y;
};

int main(void)
{

  struct punto2D punto1;
  struct punto3D punto2;

  punto1.x = 2;
  punto1.y = 3;

  punto2 = punto1;

  printf("El punto tiene coordenadas (%.1f, %.1f)\n", punto2.x, punto2.y);

  return 0;
}
```

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

struct punto3D {
  float x;
  float y;
};

int main(void)
{

  struct punto2D punto1;
  struct punto3D punto2;

  punto1.x = 2;
  punto1.y = 3;

  punto2 = punto1;

  printf("El punto tiene coordenadas (%.1f, %.1f)\n", punto2.x, punto2.y);

  return 0;
}
```

```
struct-03.c: In function 'main':
struct-03.c:22:10: error: incompatible types when assigning to type 'struct
     punto3D' from type 'struct punto2D'
   punto2 = punto1;
          ^
```

```c
#include <stdio.h>
#include <math.h>

struct punto2D {
  float x;
  float y;
};

int main(void)
{

  float distancia;

  struct punto2D p1, p2;
  struct punto2D resta;

  printf("Ingrese las coordenadas del punto 1: "); scanf("%f %f", &p1.x, &p1.y);
  printf("Ingrese las coordenadas del punto 2: "); scanf("%f %f", &p2.x, &p2.y);

  resta.x = p1.x - p2.x;
  resta.y = p1.y - p2.y;

  distancia = sqrt(resta.x*resta.x + resta.y*resta.y);

  printf("La distancia entre los dos puntos es %f\n", distancia);

  return 0;
}
```

```c
#include <stdio.h>
#include <math.h>

struct punto2D {
  float x;
  float y;
};

int main(void)
{

  float distancia;

  struct punto2D p1, p2;
  struct punto2D resta;

  printf("Ingrese las coordenadas del punto 1: "); scanf("%f %f", &p1.x, &p1.y);
  printf("Ingrese las coordenadas del punto 2: "); scanf("%f %f", &p2.x, &p2.y);

  resta.x = p1.x - p2.x;
  resta.y = p1.y - p2.y;

  distancia = sqrt(resta.x*resta.x + resta.y*resta.y);

  printf("La distancia entre los dos puntos es %f\n", distancia);

  return 0;
}
```

```
Ingrese las coordenadas del punto 1: 1 0
Ingrese las coordenadas del punto 2: 0 1
La distancia entre los dos puntos es 1.414214
```

Precedencia de Operadores

Precedencia de Operadores

Precedencia de Operadores

Operadores	Asociatividad

Precedencia de Operadores

Operadores	Asociatividad
() [] . ->	Izquierda a Derecha

Precedencia de Operadores

Operadores	Asociatividad
() [] . ->	Izquierda a Derecha
++ -- + - ! (tipo)	Derecha a Izquierda

Precedencia de Operadores

Operadores	Asociatividad
() [] . ->	Izquierda a Derecha
++ -- + - ! (tipo)	Derecha a Izquierda
* / %	Izquierda a Derecha

Precedencia de Operadores

Operadores	Asociatividad
() [] . ->	Izquierda a Derecha
++ -- + - ! (tipo)	Derecha a Izquierda
* / %	Izquierda a Derecha
+ -	Izquierda a Derecha

Precedencia de Operadores

Operadores	Asociatividad
`()` `[]` `.` `->`	Izquierda a Derecha
`++` `--` `+` `-` `!` `(tipo)`	Derecha a Izquierda
`*` `/` `%`	Izquierda a Derecha
`+` `-`	Izquierda a Derecha
`<` `<=` `>` `>=`	Izquierda a Derecha

Precedencia de Operadores

Operadores	Asociatividad
`()` `[]` `.` `->`	Izquierda a Derecha
`++` `--` `+` `-` `!` `(tipo)`	Derecha a Izquierda
`*` `/` `%`	Izquierda a Derecha
`+` `-`	Izquierda a Derecha
`<` `<=` `>` `>=`	Izquierda a Derecha
`==` `!=`	Izquierda a Derecha

Precedencia de Operadores

Operadores	Asociatividad
`()` `[]` `.` `->`	Izquierda a Derecha
`++` `--` `+` `-` `!` `(tipo)`	Derecha a Izquierda
`*` `/` `%`	Izquierda a Derecha
`+` `-`	Izquierda a Derecha
`<` `<=` `>` `>=`	Izquierda a Derecha
`==` `!=`	Izquierda a Derecha
`&&` `||`	Izquierda a Derecha

Precedencia de Operadores

Operadores						Asociatividad
()	[ ]	.	->			Izquierda a Derecha
++	--	+	-	!	(tipo)	Derecha a Izquierda
*	/	%				Izquierda a Derecha
+	-					Izquierda a Derecha
<	<=	>	>=			Izquierda a Derecha
==	!=					Izquierda a Derecha
&&	||					Izquierda a Derecha
=	+=	-=	*=	/=	%=	Derecha a Izquierda

Precedencia de Operadores

Operadores	Asociatividad
`()` `[]` `.` `->`	Izquierda a Derecha
`++` `--` `+` `-` `!` `(tipo)`	Derecha a Izquierda
`*` `/` `%`	Izquierda a Derecha
`+` `-`	Izquierda a Derecha
`<` `<=` `>` `>=`	Izquierda a Derecha
`==` `!=`	Izquierda a Derecha
`&&` `||`	Izquierda a Derecha
`=` `+=` `-=` `*=` `/=` `%=`	Derecha a Izquierda
`,`	Izquierda a Derecha

Precedencia de Operadores

Operadores	Asociatividad
`()` `[]` `.` `->`	Izquierda a Derecha
`++` `--` `+` `-` `!` `(tipo)`	Derecha a Izquierda
`*` `/` `%`	Izquierda a Derecha
`+` `-`	Izquierda a Derecha
`<` `<=` `>` `>=`	Izquierda a Derecha
`==` `!=`	Izquierda a Derecha
`&&` `||`	Izquierda a Derecha
`=` `+=` `-=` `*=` `/=` `%=`	Derecha a Izquierda
`,`	Izquierda a Derecha

```c
#include <stdio.h>

struct alumno {
  char nombre[80];
  char apellido[80];
  unsigned int legajo;
  int notas_info[3];
  int estado_info;
};

int main(void)
{
  struct alumno a1;
  int i, j , n;

  printf("Ingrese datos de alumno 1\n");
  printf("Legajo: "); scanf("%d", &a1.legajo);
  printf("Nombre: "); scanf("%s", a1.nombre);
  printf("Apellido: "); scanf("%s", a1.apellido);
  printf("Parcial 1: "); scanf("%d", &a1.notas_info[0]);
  printf("Parcial 2: "); scanf("%d", &a1.notas_info[1]);
  printf("Recuperatorio: "); scanf("%d", &a1.notas_info[2]);

  printf("Legajo\tApellido\t\tP1\tP2\t R");
  printf("-----------------------------------------");
  printf("\n%5d\t%-10.10s\t\t%2d\t%2d\t%2d\n", a1.legajo, a1.apellido,
                                              a1.notas_info[0], a1.notas_info
                                                    [1], a1.notas_info[2]);

  return 0;
}
```

```
Ingrese datos de alumno 1
Legajo: 48636
Nombre: Claudio
Apellido: Paz
Parcial 1: 2
Parcial 2: 4
Recuperatorio: 2
Legajo      Apellido           P1      P2        R
-----------------------------------------------
48636       Paz                 2       4        2
```

Alias con typedef

```
typedef unsigned int uint;
```

Alias con `typedef`

- `typedef` es la palabra clave para indicarle al compilador que se generará un *alias*

```
typedef unsigned int uint;
```

Alias con `typedef`

```
typedef unsigned int uint;
```

- `typedef` es la palabra clave para indicarle al compilador que se generará un *alias*
- En el ejemplo se hace que el alias `uint` signifique `unsigned int`. A partir de ese momento se pueden usar indistintamente.

Alias con `typedef`

```
typedef unsigned int uint;
```

- `typedef` es la palabra clave para indicarle al compilador que se generará un *alias*
- En el ejemplo se hace que el alias `uint` signifique `unsigned int`. A partir de ese momento se pueden usar indistintamente.
- Se pueden hacer alias con cualquier tipo, con el objetivo de escribir código más compacto y legible.

Alias con typedef

```c
#include <stdio.h>

typedef unsigned int uint;
typedef long long int llint;
typedef unsigned char byte;

int main(void)
{

  uint a;
  lint b;
  llint c;

  byte b1;

  /* código */


  return 0;
}
```

```c
#include <stdio.h>

struct punto2D {
  float x;
  float y;
};

typedef struct punto2D Punto2D;

int main(void)
{

  Punto2D p1;

  p1.x = 2;
  p1.y = 3;

  printf("El punto tiene coordenadas (%.2f,%.2f)\n", p1.x, p1.y);

  return 0;
}
```

```
El punto tiene coordenadas (2.00,3.00)
```

```c
#include <stdio.h>

typedef struct punto2D {
  float x;
  float y;
} Punto2D;



int main(void)
{

  Punto2D p1;

  p1.x = 2;
  p1.y = 3;

  printf("El punto tiene coordenadas (%.2f,%.2f)\n", p1.x, p1.y);

  return 0;
}
```

```
El punto tiene coordenadas (2.00,3.00)
```

```c
#include <stdio.h>

typedef struct {
  float x;
  float y;
} Punto2D;



int main(void)
{

  Punto2D p1;

  p1.x = 2;
  p1.y = 3;

  printf("El punto tiene coordenadas (%.2f,%.2f)\n", p1.x, p1.y);

  return 0;
}
```

```
El punto tiene coordenadas (2.00,3.00)
```

Consultas

claudiojpaz@gmail.com

Horario de Consulta: Martes 18:00-19:00hs
Of.5 Ed.Salcedo