Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar

C programozási nyelv
Struktúrák és Unionok

Dr. Schuster György

2017. december 29.

Struktúrák

A struktúra egy olyan összetett adatszerkezet, amely **nemcsak azonos típusú elemeket** rendelhet össze.

Struktúrák

A struktúra egy olyan összetett adatszerkezet, amely **nemcsak azonos típusú elemeket** rendelhet össze.
A struktúra definíciójában leírjuk, hogy a struktúra hogyan épül fel.

Struktúrák

A struktúra egy olyan összetett adatszerkezet, amely **nemcsak azonos típusú elemeket** rendelhet össze.
A struktúra definíciójában leírjuk, hogy a struktúra hogyan épül fel.
Például:

```
struct ember
  {
  char nev[80];
  unsigned kor;
  float suly;
  };
```

Definiáltunk egy olyan elemet, amely egyben tartalmazza egy egyén adatait.

Struktúrák

A struktúra egy olyan összetett adatszerkezet, amely **nemcsak azonos típusú elemeket** rendelhet össze.
A struktúra definíciójában leírjuk, hogy a struktúra hogyan épül fel.
Például:

```
struct ember
  {
  char nev[80];
  unsigned kor;
  float suly;
  };
```

Definiáltunk egy olyan elemet, amely egyben tartalmazza egy egyén adatait.

A deklarációnál egy adott példányt hozunk létre.

Struktúrák

A struktúra egy olyan összetett adatszerkezet, amely **nemcsak azonos típusú elemeket** rendelhet össze.
A struktúra definíciójában leírjuk, hogy a struktúra hogyan épül fel.
Például:

```
struct ember
  {
  char nev[80];
  unsigned kor;
  float suly;
  };
```

Definiáltunk egy olyan elemet, amely egyben tartalmazza egy egyén adatait.

A deklarációnál egy adott példányt hozunk létre. Példa:

```
struct ember jozsi;
```

Józsi (`jozsi`) már egy konkrét személy.

Mezőhozzáférés közvetlen és indirekt

Egy struktúra ”példány” adott mezőjéhez a mezőhozzáférés operátor ’.’ segítségével férünk hozzá.

```
jozsi.kor=43;
```

Mezőhozzáférés közvetlen és indirekt

Egy struktúra ”példány” adott mezőjéhez a mezőhozzáférés operátor ’.’ segítségével férünk hozzá.

```
jozsi.kor=43;
```

Vagyis a `jozsi` példány `kor` mezője legyen egyenlő 43-al. Ezután úgy használjuk, mint egy normális változót.

Mezőhozzáférés közvetlen és indirekt

Egy struktúra ”példány” adott mezőjéhez a mezőhozzáférés operátor ’.’ segítségével férünk hozzá.

```
jozsi.kor=43;
```

Vagyis a `jozsi` példány `kor` mezője legyen egyenlő 43-al. Ezután úgy használjuk, mint egy normális változót.

Ha nem a példány neve, hanem a címe van megadva ...

```
struct ember jozsi;
struct ember *jp;
    ⋮
jp=&jozsi;
    ⋮
jp->kor=43;
```

Mezőhozzáférés közvetlen és indirekt

Egy struktúra ”példány” adott mezőjéhez a mezőhozzáférés operátor ’.’ segítségével férünk hozzá.

```
jozsi.kor=43;
```

Vagyis a `jozsi` példány `kor` mezője legyen egyenlő 43-al. Ezután úgy használjuk, mint egy normális változót.

Ha nem a példány neve, hanem a címe van megadva ...

```
struct ember jozsi;
struct ember *jp;
    :
jp=&jozsi;
    :
jp->kor=43;
```

Szó szerint!

A `jp` által megcímzett struktúra `kor` mezője legyen egyenlő 43-al.

Rekurzív struktúra

Kérdés: mit tartalmazhat egy struktúra?

Rekurzív struktúra

Kérdés: mit tartalmazhat egy struktúra?
Válasz: bármilyen adatelemet, de **nem tartalmazhat olyan struktúrát, mint önmaga, se közvetlenül se közvetetten.**

Rekurzív struktúra

Kérdés: mit tartalmazhat egy struktúra?
Válasz: bármilyen adatelemet, de **nem tartalmazhat olyan struktúrát, mint önmaga, se közvetlenül se közvetetten.**
Viszont tartalmazhat olyan ”típusú” pointert, mint önmaga.

Rekurzív struktúra

Kérdés: mit tartalmazhat egy struktúra?
Válasz: bármilyen adatelemet, de **nem tartalmazhat olyan struktúrát, mint önmaga, se közvetlenül se közvetetten.**
Viszont tartalmazhat olyan ”típusú” pointert, mint önmaga.
Ilyen jellegű alkalmazás az úgynevezett lista adatszekezet. Egy ilyen struktúra definíciója:

```
struct LIST
  {
  int v;
  struct LIST *next;
  };
```

Rekurzív struktúra

Kérdés: mit tartalmazhat egy struktúra?
Válasz: bármilyen adatelemet, de **nem tartalmazhat olyan struktúrát, mint önmaga, se közvetlenül se közvetetten.**
Viszont tartalmazhat olyan ”típusú” pointert, mint önmaga.
Ilyen jellegű alkalmazás az úgynevezett lista adatszekezet. Egy ilyen struktúra definíciója:

```
struct LIST
  {
  int v;
  struct LIST *next;
  };
```

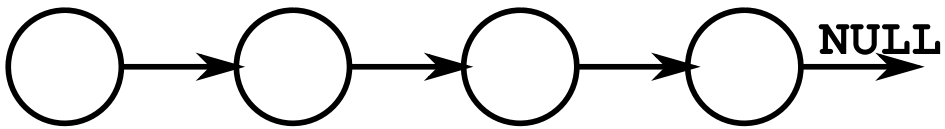
A `v` az adatelem, a `next` az a pointer, amely a következő elemre mutat.

Rekurzív struktúra

Kérdés: mit tartalmazhat egy struktúra?
Válasz: bármilyen adatelemet, de **nem tartalmazhat olyan struktúrát, mint önmaga, se közvetlenül se közvetetten.**
Viszont tartalmazhat olyan ”típusú” pointert, mint önmaga.
Ilyen jellegű alkalmazás az úgynevezett lista adatszekezet. Egy ilyen struktúra definíciója:

```
struct LIST
  {
  int v;
  struct LIST *next;
  };
```

A `v` az adatelem, a `next` az a pointer, amely a következő elemre mutat. Valahogy így.



Típusként definiált struktúra

A struktúrákat a `typedef` utasítással is definiálhatjuk.

Típusként definiált struktúra

A struktúrákat a `typedef` utasítással is definiálhatjuk.
Példa:

```
typedef struct
  {
  char nev[80];
  unsigned kor;
  float suly;
  } ember;
```

Típusként definiált struktúra

A struktúrákat a `typedef` utasítással is definiálhatjuk.
Példa:

```
typedef struct
  {
  char nev[80];
  unsigned kor;
  float suly;
  } ember;
```

Ettől kezdve van egy `ember` típusunk. Láthatjuk, hogy a mezők azonosak az előzőekben ”hagyományos” módon definiált struktúránál.

Típusként definiált struktúra

A struktúrákat a `typedef` utasítással is definiálhatjuk.
Példa:

```
typedef struct
  {
  char nev[80];
  unsigned kor;
  float suly;
  } ember;
```

Ettől kezdve van egy `ember` típusunk. Láthatjuk, hogy a mezők azonosak az előzőekben ”hagyományos” módon definiált struktúránál.

A deklaráció egyszerűsödik.

```
ember jozsi;
```

Típusként definiált struktúra

A struktúrákat a `typedef` utasítással is definiálhatjuk.
Példa:

```
typedef struct
  {
  char nev[80];
  unsigned kor;
  float suly;
  } ember;
```

Ettől kezdve van egy `ember` típusunk. Láthatjuk, hogy a mezők azonosak az előzőekben ”hagyományos” módon definiált struktúránál.

A deklaráció egyszerűsödik.

```
ember jozsi;
```

A `struct` kulcsszó elmaradt.

Típusként definiált struktúra

A struktúrákat a `typedef` utasítással is definiálhatjuk.
Példa:

```
typedef struct
  {
  char nev[80];
  unsigned kor;
  float suly;
  } ember;
```

Ettől kezdve van egy `ember` típusunk. Láthatjuk, hogy a mezők azonosak az előzőekben ”hagyományos” módon definiált struktúránál.

A deklaráció egyszerűsödik.

```
ember jozsi;
```

A `struct` kulcsszó elmaradt.

Nem lehetséges rekurzív struktúrát definiálni ezen a módon!

Típusként definiált struktúra

A struktúrákat a `typedef` utasítással is definiálhatjuk.
Példa:

```
typedef struct
  {
  char nev[80];
  unsigned kor;
  float suly;
  } ember;
```

Ettől kezdve van egy `ember` típusunk. Láthatjuk, hogy a mezők azonosak az előzőekben ”hagyományos” módon definiált struktúránál.

A deklaráció egyszerűsödik.

```
ember jozsi;
```

A `struct` kulcsszó elmaradt.

Nem lehetséges rekurzív struktúrát definiálni ezen a módon!
Miért?
A típus a struktúra után derül ki.
Amúgy minden más azonos.

Bitmezők

Egész jellegű változók esetén definiálhatók az un. bitmezők.
Összehasonlító példa:

```
struct chr
   {
   unsigned x;
   unsigned y;
   char code;
   };
```

Ez minimum 9 bájt.

Bitmezők

Egész jellegű változók esetén definiálhatók az un. bitmezők.
Összehasonlító példa:

```
struct chr
   {
   unsigned x;
   unsigned y;
   char code;
   };
```

Ez minimum 9 bájt.

```
struct chr
   {
   unsigned x:  5;
   unsigned y:  7;
   unsigned code:  8;
   };
```

Ez 20 bit, de nem lehet az alaptípusnál rövidebb, tehát 4 bájt.

Bitmezők

Egész jellegű változók esetén definiálhatók az un. bitmezők.
Összehasonlító példa:

```
struct chr
   {
   unsigned x;
   unsigned y;
   char code;
   };
```

Ez minimum 9 bájt.

```
struct chr
   {
   unsigned x:  5;
   unsigned y:  7;
   unsigned code:  8;
   };
```

Ez 20 bit, de nem lehet az alaptípusnál rövidebb, tehát 4 bájt.

Ne feledjük, hogy a bitek ”tologatása” növeli a futásidőt.

Bitmezők

Egész jellegű változók esetén definiálhatók az un. bitmezők.
Összehasonlító példa:

```
struct chr
   {
   unsigned x;
   unsigned y;
   char code;
   };
```

Ez minimum 9 bájt.

```
struct chr
   {
   unsigned x:  5;
   unsigned y:  7;
   unsigned code:  8;
   };
```

Ez 20 bit, de nem lehet az alaptípusnál rövidebb, tehát 4 bájt.

Ne feledjük, hogy a bitek ”tologatása” növeli a futásidőt.
PC-s környezetben csak a hardver programozására használjuk.

Bitmezők

Egész jellegű változók esetén definiálhatók az un. bitmezők.
Összehasonlító példa:

```
struct chr
   {
   unsigned x;
   unsigned y;
   char code;
   };
```

Ez minimum 9 bájt.

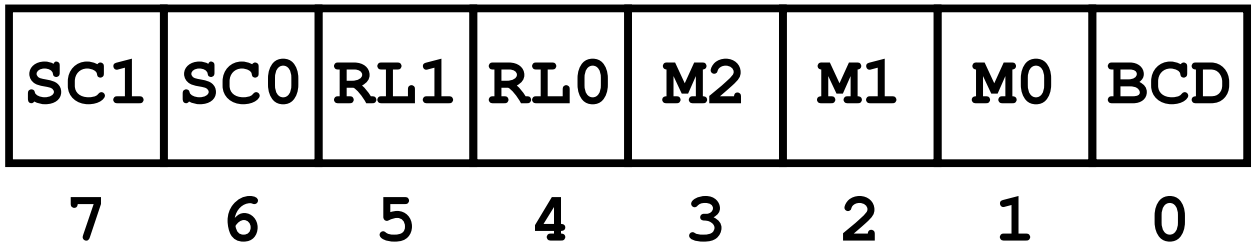
```
struct chr
   {
   unsigned x:  5;
   unsigned y:  7;
   unsigned code:  8;
   };
```

Ez 20 bit, de nem lehet az alaptípusnál rövidebb, tehát 4 bájt.

Ne feledjük, hogy a bitek ”tologatása” növeli a futásidőt.
PC-s környezetben csak a hardver programozására használjuk.
Kontrollereknél helytakarékossági szempontokból is használjuk.

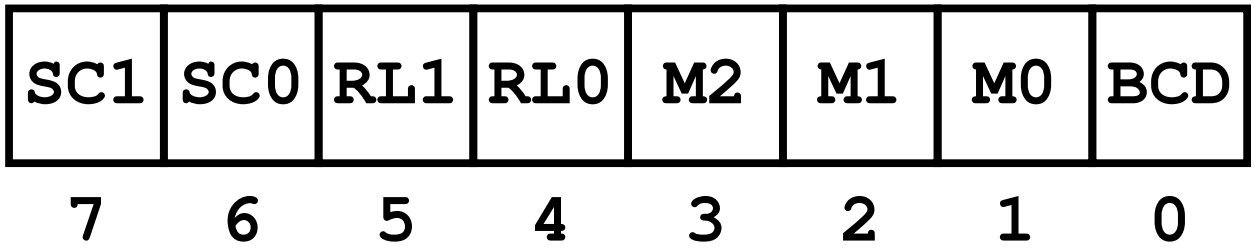
Bitmező hardver példa

Az Intel 8254 timer IC parancs regisztere:



Bitmező hardver példa

Az Intel 8254 timer IC parancs regisztere:




```
typedef struct
   {
   unsigned char bcd:  1;
   unsigned char m:  3;
   unsigned char rl:  2;
   unsigned char sc:  2;
   } c8254;
```

union

Speciális szerkezet. Arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.

union

Speciális szerkezet. Arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.

Példa:

Definíció:

```
union smpl
   {
   float a;
   int b;
   };
```

union

Speciális szerkezet. Arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.
Példa:

Definíció:

```
union smpl
   {
   float a;
   int b;
   };
```

Deklaráció:

```
      ⋮
union smpl minta;
      ⋮
```

union

Speciális szerkezet. Arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.

Példa:

Definíció:

```
union smpl
  {
  float a;
  int b;
  };
```

Deklaráció:

```
    :
union smpl minta;
    :
```

A mezőhozzáférés ugyanúgy történik, mint a struktúráknál:

```
minta.a=3.14159265;
```

union

Speciális szerkezet. Arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.

Példa:

Definíció:

```
union smpl
  {
  float a;
  int b;
  };
```

Deklaráció:

```
    ⋮
union smpl minta;
    ⋮
```

A mezőhozzáférés ugyanúgy történik, mint a struktúráknál:

```
minta.a=3.14159265;
```

Ha az `a` mezővel hivatkozunk a terület `float`-ként lesz kezelve.

union

Speciális szerkezet. Arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.

Példa:

Definíció:

```
union smpl
   {
   float a;
   int b;
   };
```

Deklaráció:

```
      ⋮
union smpl minta;
      ⋮
```

A mezőhozzáférés ugyanúgy történik, mint a struktúráknál:

```
minta.a=3.14159265;
```

Ha az `a` mezővel hivatkozunk a terület `float`-ként lesz kezelve.

Ha a `b` mezővel hivatkozunk a terület `int`-ként lesz kezelve.

union

Speciális szerkezet. Arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.

Példa:

Definíció:

```
union smpl
   {
   float a;
   int b;
   };
```

Deklaráció:

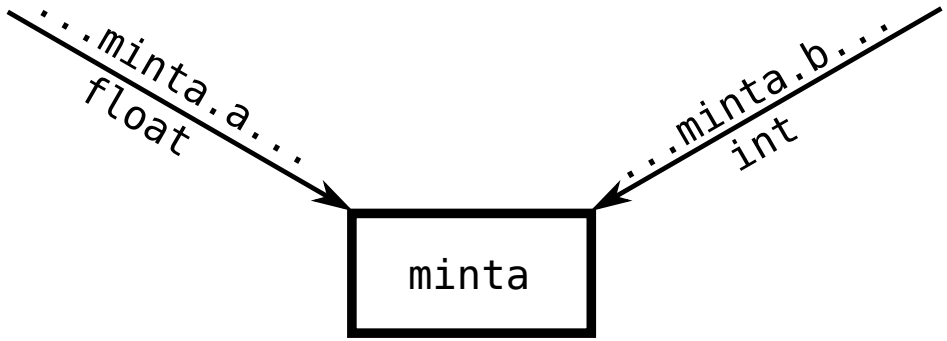
```
      ⋮
union smpl minta;
      ⋮
```

A mezőhozzáférés ugyanúgy történik, mint a struktúráknál:

```
minta.a=3.14159265;
```

Ha az **a** mezővel hivatkozunk a terület **float**-ként lesz kezelve.

Ha a **b** mezővel hivatkozunk a terület **int**-ként lesz kezelve.



Funkcionális példa

Feladat egy `float` adatot kell egyik számítógépről egy másikra átküldeni, de az adatcsatorna csak bájtokat képes továbbítani.

Funkcionális példa

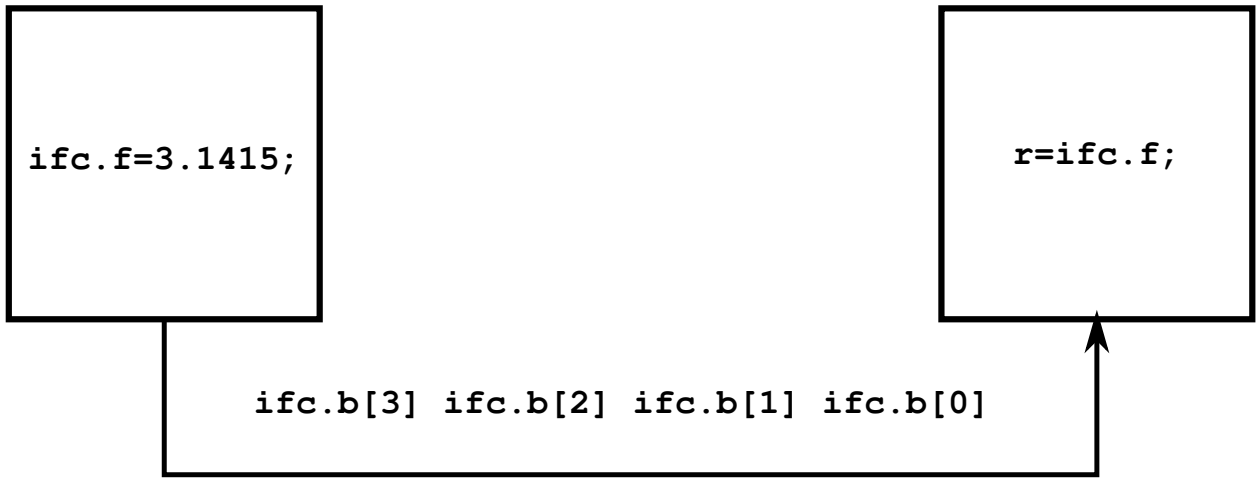
Feladat egy `float` adatot kell egyik számítógépről egy másikra átküldeni, de az adatcsatorna csak bájtokat képes továbbítani. Célszerű egy uniont-t használni.

```
union IFC
  {
  float f;
  char b[4];
  } ifc;
```

Funkcionális példa

Feladat egy `float` adatot kell egyik számítógépről egy másikra átküldeni, de az adatcsatorna csak bájtokat képes továbbítani. Célszerű egy uniont-t használni.

```
union IFC
  {
  float f;
  char b[4];
  } ifc;
```

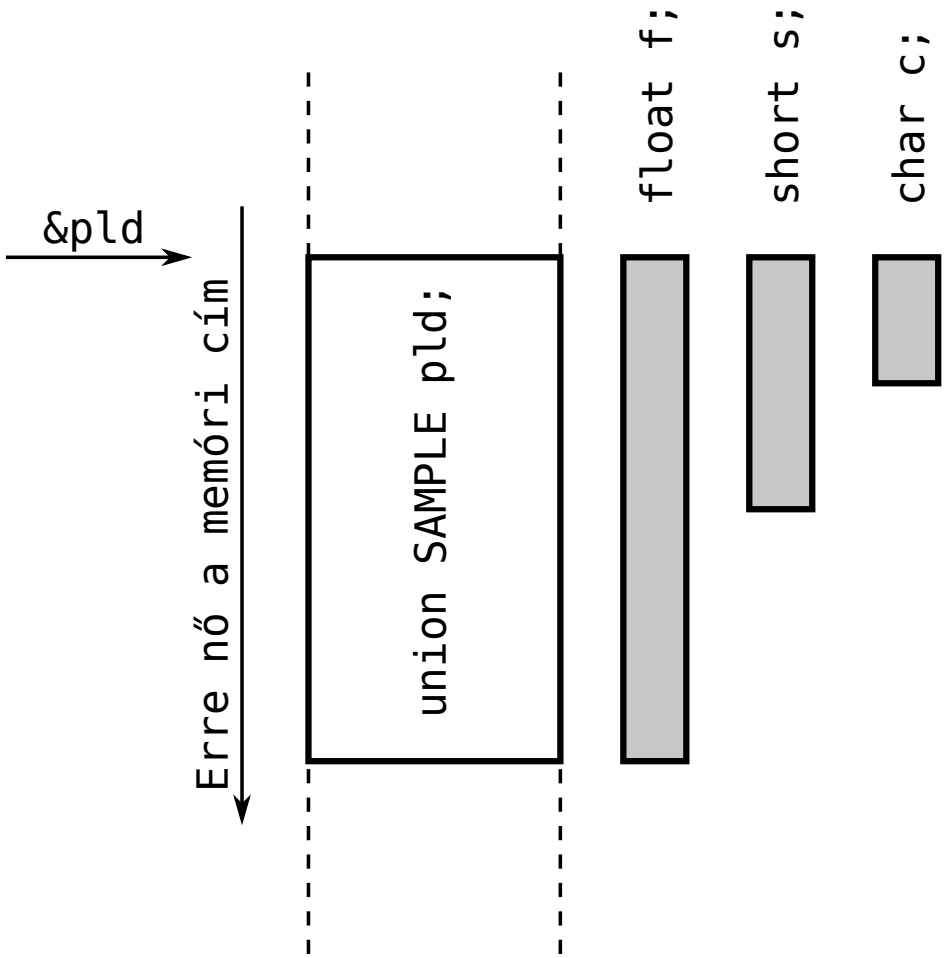


Eltérő méretű mezők

```
union SAMPLE
   {
   float f;
   short s;
   char c;
   } pld;
```

Eltérő méretű mezők

```
union SAMPLE
   {
   float f;
   short s;
   char c;
   } pld;
```



Típusként definiált union

Hagyományos módszer:

Definíció:

```
union smpl
   {
   float a;
   int b;
   };
```

Deklaráció:

```
   ⋮
union smpl pld;
   ⋮
```

Típusként definiált union

Hagyományos módszer:

Definíció:

```
union smpl
   {
   float a;
   int b;
   };
```

Deklaráció:

```
   ⋮
union smpl pld;
   ⋮
```

Típusként

Definíció:

```
typedef union
   {
   float a;
   int b;
   } smpl;
```

Deklaráció:

```
   ⋮
smpl pld;
   ⋮
```

Kérdések

Mi az a struktúra?

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: `pld.field=5;`

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: `pld.field=5;`

Milyen típusú mezők lehetnek egy struktúrában?

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: `pld.field=5;`

Milyen típusú mezők lehetnek egy struktúrában?

Bármilyen típus, módosított típus, bármilyen pointer, struktúra is. Kivétel olyan "típusú" struktúra, mint az éppen definiált.

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: `pld.field=5;`

Milyen típusú mezők lehetnek egy struktúrában?

Bármilyen típus, módosított típus, bármilyen pointer, struktúra is. Kivétel olyan "típusú" struktúra, mint az éppen definiált.

Lehet-e a strutúrákhoz pointereket rendelni?

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: **`pld.field=5;`**

Milyen típusú mezők lehetnek egy struktúrában?

Bármilyen típus, módosított típus, bármilyen pointer, struktúra is. Kivétel olyan "típusú" struktúra, mint az éppen definiált.

Lehet-e a strutúrákhoz pointereket rendelni?

Igen, például: **`struct prb *sp;`**

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: **`pld.field=5;`**

Milyen típusú mezők lehetnek egy struktúrában?

Bármilyen típus, módosított típus, bármilyen pointer, struktúra is. Kivétel olyan "típusú" struktúra, mint az éppen definiált.

Lehet-e a strutúrákhoz pointereket rendelni?

Igen, például: **`struct prb *sp;`**

Lehet-e a struktúra pointer segítségével adott struktúra példány adott mezőjét elérni?

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: **`pld.field=5;`**

Milyen típusú mezők lehetnek egy struktúrában?

Bármilyen típus, módosított típus, bármilyen pointer, struktúra is. Kivétel olyan "típusú" struktúra, mint az éppen definiált.

Lehet-e a strutúrákhoz pointereket rendelni?

Igen, például: **`struct prb *sp;`**

Lehet-e a struktúra pointer segítségével adott struktúra példány adott mezőjét elérni?

Igen a struktúra indirekciós operátorral **`->`**, példa:

`sp->field=5;`

Kérdések

Mi az a struktúra?
A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.
Mit jelent a struktúra definíciója?
Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.
Mit jelent a strutúra deklarációja?
Ez a struktúra példány létrehozása.
Hogyan érhető el egy mező az adott struktúra példányban?
A mező hozzáférés operátorral '`.`', például: **`pld.field=5;`**
Milyen típusú mezők lehetnek egy struktúrában?
Bármilyen típus, módosított típus, bármilyen pointer, struktúra is. Kivétel olyan "típusú" struktúra, mint az éppen definiált.
Lehet-e a strutúrákhoz pointereket rendelni?
Igen, például: **`struct prb *sp;`**
Lehet-e a struktúra pointer segítségével adott struktúra példány adott mezőjét elérni?
Igen a struktúra indirekciós operátorral **`->`**, példa:
`sp->field=5;`
Mit jelent a rekurzív struktúra fogalom?

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: **`pld.field=5;`**

Milyen típusú mezők lehetnek egy struktúrában?

Bármilyen típus, módosított típus, bármilyen pointer, struktúra is. Kivétel olyan "típusú" struktúra, mint az éppen definiált.

Lehet-e a strutúrákhoz pointereket rendelni?

Igen, például: **`struct prb *sp;`**

Lehet-e a struktúra pointer segítségével adott struktúra példány adott mezőjét elérni?

Igen a struktúra indirekciós operátorral **`->`**, példa:

`sp->field=5;`

Mit jelent a rekurzív struktúra fogalom?

A struktúra olyan mezőt tartalmaz, amely az adott típusú struktúra pointerét tartalmazza.

Kérdések

Mi az a struktúra?

A struktúra egy olyan összetett adatszerkezet, amely nemcsak azonos típusú elemeket rendelhet össze.

Mit jelent a struktúra definíciója?

Ebben adjuk meg az adott struktúra szerkezetét a mezők deklarálásával.

Mit jelent a strutúra deklarációja?

Ez a struktúra példány létrehozása.

Hogyan érhető el egy mező az adott struktúra példányban?

A mező hozzáférés operátorral '`.`', például: **`pld.field=5;`**

Milyen típusú mezők lehetnek egy struktúrában?

Bármilyen típus, módosított típus, bármilyen pointer, struktúra is. Kivétel olyan "típusú" struktúra, mint az éppen definiált.

Lehet-e a strutúrákhoz pointereket rendelni?

Igen, például: **`struct prb *sp;`**

Lehet-e a struktúra pointer segítségével adott struktúra példány adott mezőjét elérni?

Igen a struktúra indirekciós operátorral **`->`**, példa:

`sp->field=5;`

Mit jelent a rekurzív struktúra fogalom?

A struktúra olyan mezőt tartalmaz, amely az adott típusú struktúra pointerét tartalmazza.

Kérdések

Lehte-e típusként definiálni struktúrát?

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?
Csak egész jellegűek lehetnek és a mező nem kehet nagyobb, mint az alaptípusa a mezőnek.

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?
Csak egész jellegűek lehetnek és a mező nem kehet nagyobb, mint az alaptípusa a mezőnek.
Hogyan határozható meg egy struktúra mérete és miért?

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?
Csak egész jellegűek lehetnek és a mező nem kehet nagyobb, mint az alaptípusa a mezőnek.
Hogyan határozható meg egy struktúra mérete és miért?
A **`sizeof`** operátorral, mert a processzorok optimalizálhatják a memória használatot.

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?
Csak egész jellegűek lehetnek és a mező nem kehet nagyobb, mint az alaptípusa a mezőnek.
Hogyan határozható meg egy struktúra mérete és miért?
A **`sizeof`** operátorral, mert a processzorok optimalizálhatják a memória használatot.
Mi az a union?

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?
Csak egész jellegűek lehetnek és a mező nem kehet nagyobb, mint az alaptípusa a mezőnek.
Hogyan határozható meg egy struktúra mérete és miért?
A **`sizeof`** operátorral, mert a processzorok optimalizálhatják a memória használatot.
Mi az a union?
Az union olyan speciális adatszerkezet, amely arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?
Csak egész jellegűek lehetnek és a mező nem kehet nagyobb, mint az alaptípusa a mezőnek.
Hogyan határozható meg egy struktúra mérete és miért?
A **`sizeof`** operátorral, mert a processzorok optimalizálhatják a memória használatot.
Mi az a union?
Az union olyan speciális adatszerkezet, amely arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.
Mekkora az union mérete?

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?
Csak egész jellegűek lehetnek és a mező nem kehet nagyobb, mint az alaptípusa a mezőnek.
Hogyan határozható meg egy struktúra mérete és miért?
A **`sizeof`** operátorral, mert a processzorok optimalizálhatják a memória használatot.
Mi az a union?
Az union olyan speciális adatszerkezet, amely arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.
Mekkora az union mérete?
Amekkora a legnagyobb mező mérete.

Kérdések

Lehte-e típusként definiálni struktúrát?
Lehet a **`typedef`** segítségével.
Mit jelent a bitmező struktúra?
A struktúra egész típusú mezői adott bit szélességűekre vannak felbontva.
Milyen megkötés van a bitmezőkre?
Csak egész jellegűek lehetnek és a mező nem kehet nagyobb, mint az alaptípusa a mezőnek.
Hogyan határozható meg egy struktúra mérete és miért?
A **`sizeof`** operátorral, mert a processzorok optimalizálhatják a memória használatot.
Mi az a union?
Az union olyan speciális adatszerkezet, amely arra szolgál, hogy egy adott memóriaterületet többféleképpen érhessünk el.
Mekkora az union mérete?
Amekkora a legnagyobb mező mérete.