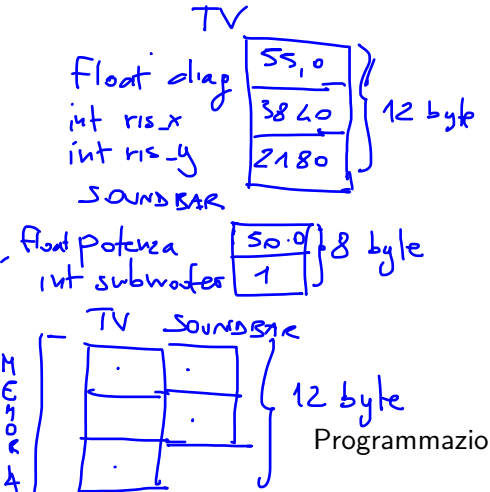


Union

Marco Alberti

Dipartimento di Matematica e Informatica

Università degli Studi di Ferrara

Programmazione e Laboratorio, A.A. 2020-2021

Ultima modifica: 27 novembre 2020

Attenzione! Questo materiale didattico è per uso personale dello studente ed è coperto da copyright.
Ne sono vietati la riproduzione e il riutilizzo anche parziale, ai sensi e per gli effetti della legge sul diritto d'autore.

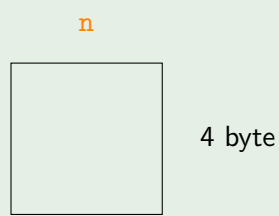
Union

⟨*tipoUnion*⟩ ::= **union {** ⟦ ⟨*tipo*⟩ ⟨*identificatore*⟩ **;** ⟧$^+$ **}**

Una union è una variabile che può contenere, in momenti diversi, dati di tipo diverso, che occupano un'area di memoria condivisa sufficientemente grande per contenere il più grande dei dati possibili.

Esempio

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

Numero n;
n.a = 3;
printf("%d", n.a);
n.b = 3.5;
printf("%f", n.b);
```



Union

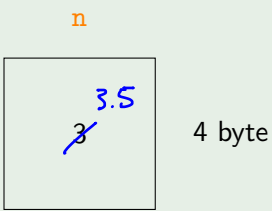
$\langle tipoUnion \rangle ::= \texttt{union \{} [\![\langle tipo \rangle \langle identificatore \rangle \texttt{ ;}]\!]^+ \texttt{\}}$

Una union è una variabile che può contenere, in momenti diversi, dati di tipo diverso, che occupano un'area di memoria condivisa sufficientemente grande per contenere il più grande dei dati possibili.

Esempio

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

Numero n;
n.a = 3;
printf("%d", n.a);
n.b = 3.5;
printf("%f", n.b);
```

stampa '3'



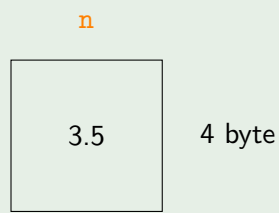
Union

$\langle tipoUnion \rangle ::=$ **union {** $[\![\langle tipo \rangle \langle identificatore \rangle$ **;** $]\!]^+$ **}**

Una union è una variabile che può contenere, in momenti diversi, dati di tipo diverso, che occupano un'area di memoria condivisa sufficientemente grande per contenere il più grande dei dati possibili.

Esempio

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

Numero n;
n.a = 3;
printf("%d", n.a);
n.b = 3.5;
printf("%f", n.b); //stampa "3.5"
```

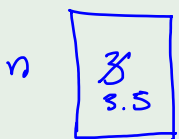


Note

- Non è necessario che tutti i campi occupino la stessa quantità di memoria, come `float` e `int`: se i campi occupano quantità diverse, la dimensione della union è pari alla maggiore delle dimensioni dei campi.
- Solo l'ultimo campo assegnato ha un valore significativo.

Esempio

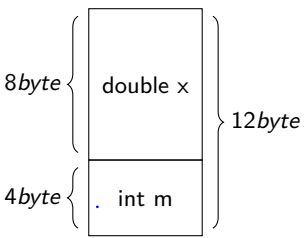
```
typedef union
  {int a;
   float b;}
Numero;

Numero n;
n.a = 3;
n.b = 3.5;
printf("%d", n.a);
```



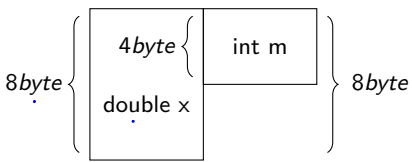
stampa 1080033280.

Struct vs. Union

```
typedef struct
  {double x;
   int m;} Numero;
```




```
typedef union
  {double x;
   int m;} Numero;
```



Come ricordare il campo significativo?

Il linguaggio non permette di riconoscere quale campo è significativo: occorre memorizzarlo, ad esempio inserendo il dato in una struttura e aggiungendo un campo enumerativo.


```
typedef struct
{
  enum
  {
    intero,
    reale
  } tipo;
  union {
    int dato_intero;
    float dato_reale;
  } dato;
} Numero;
```

```
void assegna_intero(Numero *n, int
    i)
{
  n->tipo = intero;
  n->dato.dato_intero = i;
}

void assegna_reale(Numero *n, float
    f)
{
  n->tipo = reale;
  n->dato.dato_reale = f;
}
```

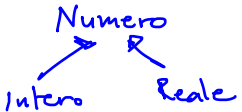
Questa tecnica simula una discriminated union, disponibile in altri linguaggi.

Utilizzo del discriminante

Le funzioni che operano sulla struttura `Numero` possono usare il discriminante per selezionare il campo significativo:


```
void stampa(Numero n)
{
  switch (n.tipo)
  {
  case (intero):
    printf("%d\n", n.dato.
        dato_intero);
    break;
  case (reale):
    printf("%f\n", n.dato.
        dato_reale);
      break;
  }
}
```

```
int main()
{
  Numero n;
  assegna_intero(&n, 3);
  stampa(n);
  assegna_reale(&n, 3.5);
  stampa(n);
  return 0;
}
```



Esercizio

Figure

Si scriva un programma che definisca un tipo `figura` in grado di rappresentare le seguenti quattro figure geometriche:

- Quadrato (caratterizzato dal lato)
- Cerchio (caratterizzato dal raggio)
- Rettangolo (caratterizzato da base e altezza)
- Triangolo (caratterizzato dai tre lati)

PERIMETRO AREA

r 6.28*r 3.14*r*r

a+b+c $p = \frac{a+b+c}{2}$ $\sqrt{p(p-a)(p-b)(p-c)}$

a b c

e che calcoli l'area e il perimetro di una figura con due funzioni aventi rispettivamente la seguente interfaccia:

- `float area(figura f);`
- `float perimetro(figura f);`