

Union

Marco Alberti

Programmazione e Laboratorio, A.A. 2016-2017

Dipartimento di Matematica e Informatica - Università di Ferrara

Ultima modifica: 22 novembre 2016

Union

$\langle \text{tipoUnion} \rangle ::= \text{'union' '{' } \{ \langle \text{tipo} \rangle \langle \text{identificatore} \rangle \text{';' } \} \text{'}};$

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;
```

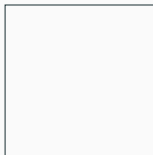
```
n.a = 3;
```

```
printf("%d", n.a);
```

```
n.b = 3.5;
```

```
printf("%f", n.b);
```

n



4 byte

Union

$\langle \text{tipoUnion} \rangle ::= \text{'union' '}' \{ \langle \text{tipo} \rangle \langle \text{identificatore} \rangle \text{';' } \} \text{'}'};$

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;
```

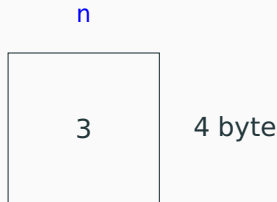
```
numero n;
```

```
n.a = 3;
```

```
printf("%d", n.a);
```

```
n.b = 3.5;
```

```
printf("%f", n.b);
```



Union

$\langle \text{tipoUnion} \rangle ::= \text{'union' '{' } \{ \langle \text{tipo} \rangle \langle \text{identificatore} \rangle \text{';' } \} \text{'}};$

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);
```

n



4 byte

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;
```

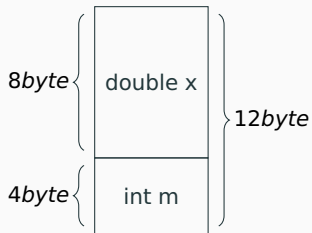
```
n.b = 3.5;
```

```
printf("%d", n.a);
```

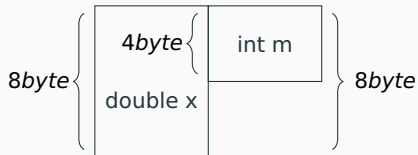
stampa 1080033280.

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;
}
```

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)

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);`