Langage C

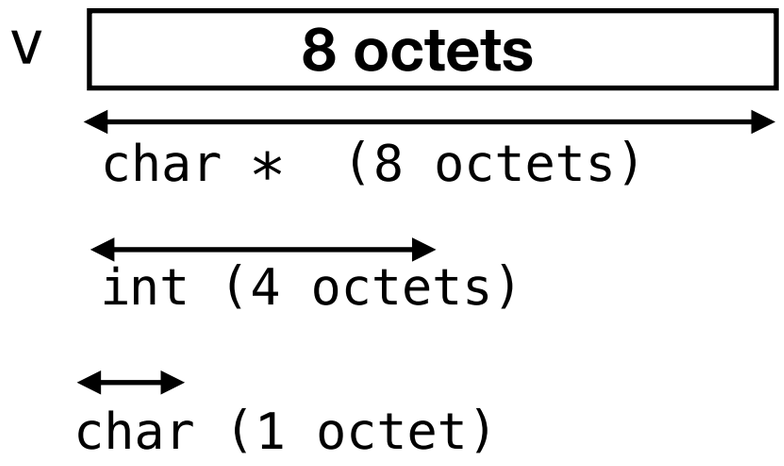
Wieslaw Zielonka
zielonka@irif.fr

union versus struct

union

Permet à stocker dans une variables une seule valeurs parmi plusieurs valeurs de types différents:

```
union noeud{
  char operation;
  int  nombre;
  char *variable;
};
union noeud v;
```



`sizeof( union noeud )` est le maximum de sizeof de champs de l'union.

v permet de stocker

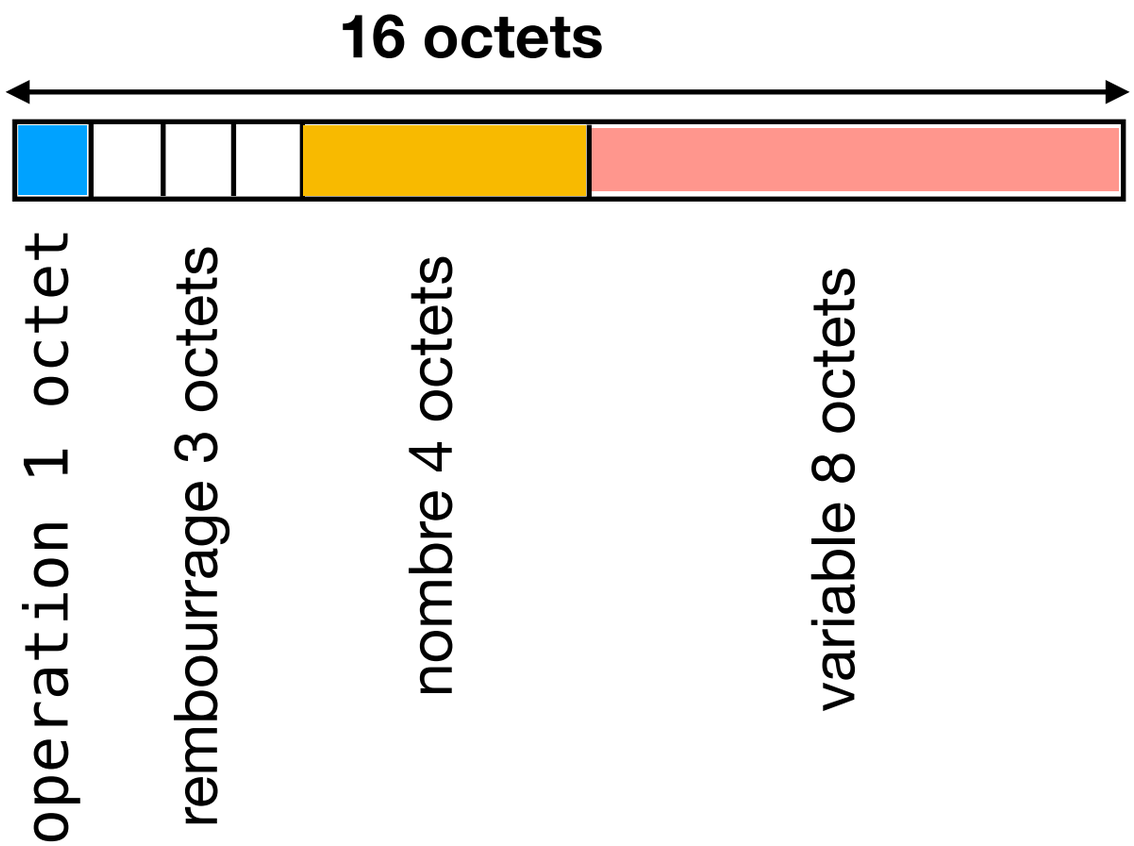
- soit un char
- soit un int
- soit un pointeur

structure

Permet à stocker dans une variables plusieurs valeurs de types différents:

```
struct struct{
  char operation;
  int  nombre;
  char *variable;
};
struct noeud w;
```

w permet de stocker 3 valeurs (char, int, char *) en même temps



les unions

```
union noeud{
  char operation;  int nombre;  char *variable;
};
```

/* déclaration et initialisation d'une variable w de type union initialisation */

```
union noeud w = { .nombre = 3 };
```

```
int a = w.nombre * 2;
```

```
w.variable = "toto";   /*w stocke désormais un pointeur*/
```

Est-ce que C "sait" si w contient un char, un int ou un pointeur char * ?

Réponse : no. C'est à nous de savoir ce que contient la variable w.

les unions -- comment savoir quel champ est valide dans une union ?

```
union valeur{
  char   op;
  double nbr_double;
  int    nbr_int;
  char * var;
};

enum type_donnee{ OPERATION, NOMBRE_DOUBLE, NOMBRE_INT, VARIABLE };

struct noeud{
  enum type_donnee t;
  union valeur     val;
};
struct noeud n = { .t = NOMBRE_DOUBLE,

                   .val = { . nbr_double = 3.14 } };

printf("%f\n",   n.val.nbr_double);

n.t = OPERATION; n.val.op = '+';
```

pointeur vers union

```
union valeur{
  char op;
  double nbr;
  char *var;
};

union valeur *v = malloc(sizeof(union valeur));
if( v == NULL ){
    /* traiter erreur de malloc */
}
v->var = "toto";    //même notation que pour les pointeurs de structures
```

structure avec un tableau de taille 0

tableau de longueur 0 dans une structure

```
typedef struct{
  unsigned char nb_elem;
  unsigned char capacity;
  int tab[]; /* tableau de 0 éléments */
} tabdyn;
```

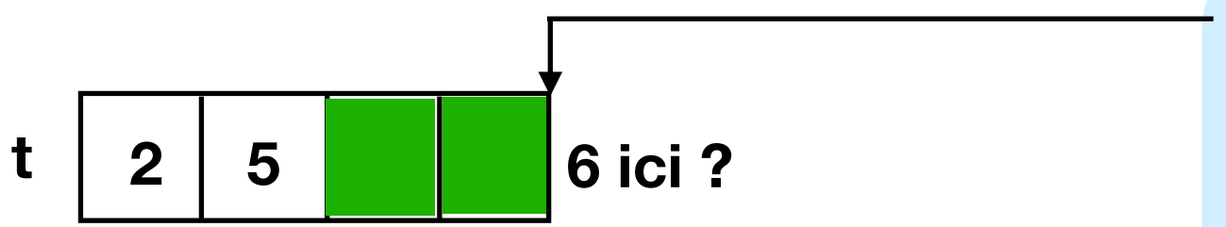
Le dernier champ d'une structure peut être un tableau de 0 éléments.

Quelle utilité ?

Aucune si on utilise une variable de type tabdyn :

```
tabdyn t;
t.nb_elem = 2;
t.capacity = 5;
t.tab[0] = 6;
```

incorrect! la variable **t** possède la mémoire pour les champs **nb_elem** et **capacity** mais pas pour le champ **tab**



ici commence le champ tab mais il a 0 octets, tentative de mettre 6 à l'adresse qui n'appartient pas à la variable t

2 octets de rembourrage pour que l'adresse de tab soit alignée sur un multiple de sizeof(int)

tableau de longueur 0 dans une structure pour un tableau dynamique

```
typedef struct{
  unsigned char nb_elem;
  int tab[]; /* tableau de 0 éléments */
} tabdyn;

tabdyn *pt = malloc( sizeof(tabdyn) + 10 * sizeof( int ) );
pt->nb_elem = 10;
for( int i = 0; i < 10; i++ ){
     pt->tab[ i ] = i*i + 1;
}
```

la mémoire allouée pour le champ tab