heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

Analyse et programmation 2

Compléments sur les types de données

(union, struct, etc.)

Thèmes abordés

- Les unions.
- Les énumérations.
- Les champs de bits.
- Concevoir un type de données.

Les types énumérés

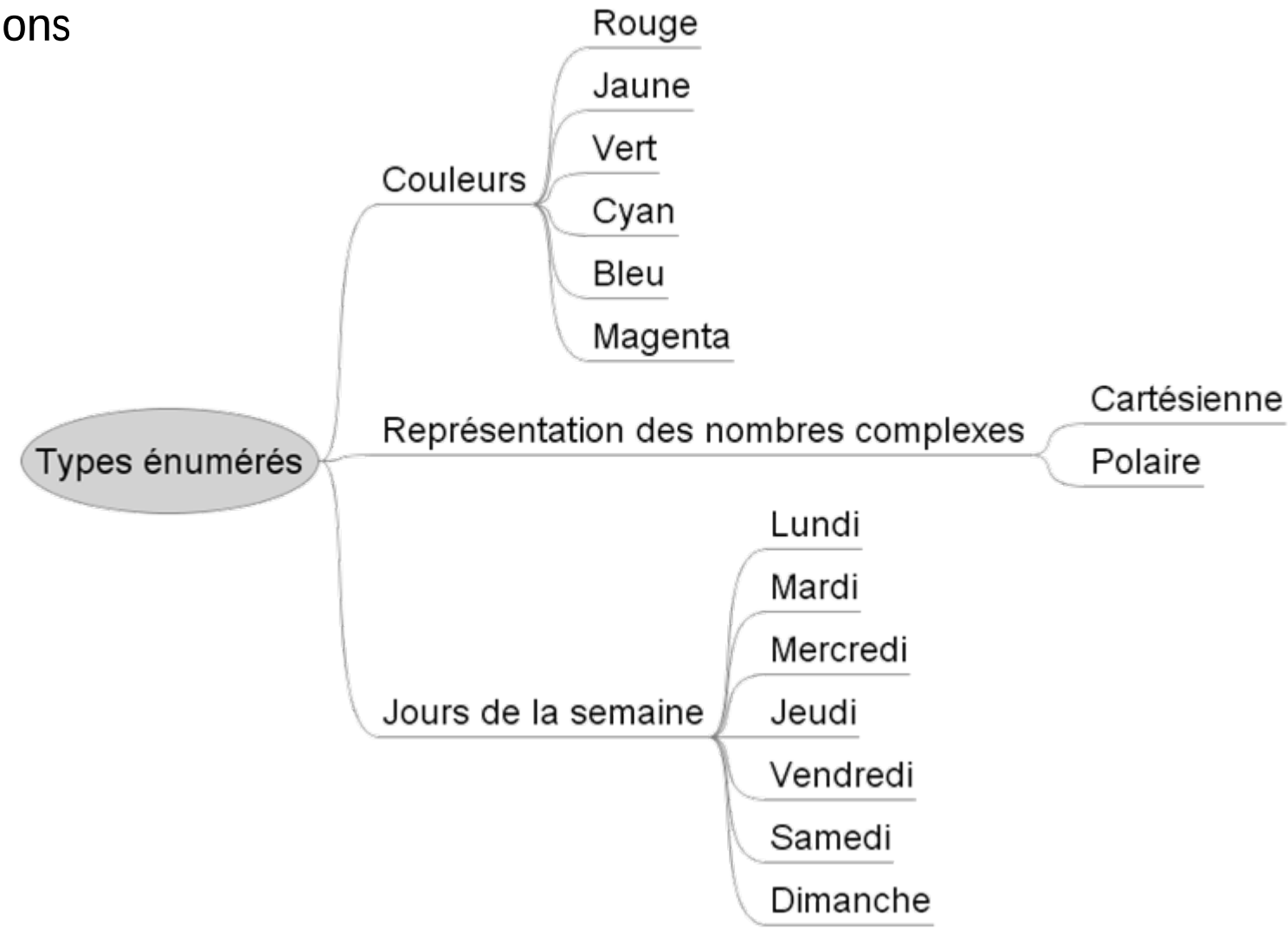
Applications

- Grandeurs ayant un nombre fini de valeurs possibles

- Exemples dans le monde réel?

Les types énumérés

Applications



Les types énumérés

Déclaration

- Forme de base

```
enum COULEUR { rouge, jaune, vert, cyan,
  bleu, magenta } une_couleur;

enum COULEUR couleur1;
couleur1 = rouge;
```

- Création d'un type avec typedef

```
typedef enum { rouge, jaune, vert, cyan,
  bleu, magenta } COULEUR;

COULEUR couleur1;
couleur1 = rouge;
```

Les types énumérés

Représentation en mémoire

- Valeurs
  - Un type énuméré est représenté comme un int
  - La première valeur de l'énumération vaut 0.
  - Les suivantes prennent les valeurs successives.
- Forçage des valeurs

```
enum COULEUR { rouge= 1, jaune, vert, cyan,
  bleu = 12, magenta } une_couleur;
```

  - jaune vaut 2, vert vaut 3, magenta vaut 13

Les types énumérés

Analyse

- Avantages
  - Plus parlant que d'utiliser un entier pour les énumérations.
- Faiblesses
  - En C, une énumération est équivalente à un entier.

```
couleur2 = bleu + magenta;
// couleur2 vaut 25, qui n'est pas dans l'énumération
```

Les unions

Des variables multiformes

- Définition
  - Une union comporte plusieurs champs.
  - Tous les champs occupent le même emplacement mémoire.
- Déclaration
  - Les mêmes formes que struct.
  - Utiliser le mot union à la place de struct.

Les unions

Des variables multiformes

- Exemple

```
typedef union
{
    char _8bits;
    short _16bits;
    long _32bits;
} ENTIER32BITS; // contient un entier 8, 16 ou 32 bits
```

- Taille

```
printf("%d\n", sizeof(ENTIER32BITS));// Affiche 4
```

- Utilisation

```
ENTIER32BITS e1;
e1._32bits = 1000;
printf("%ld\n", e1._32bits); // Affiche 1000
e1._8bits = 50;
printf("%ld\n", e1._8bits);  // Affiche 50
printf("%ld\n", e1._32bits); // Affiche 818 !
```

Les unions

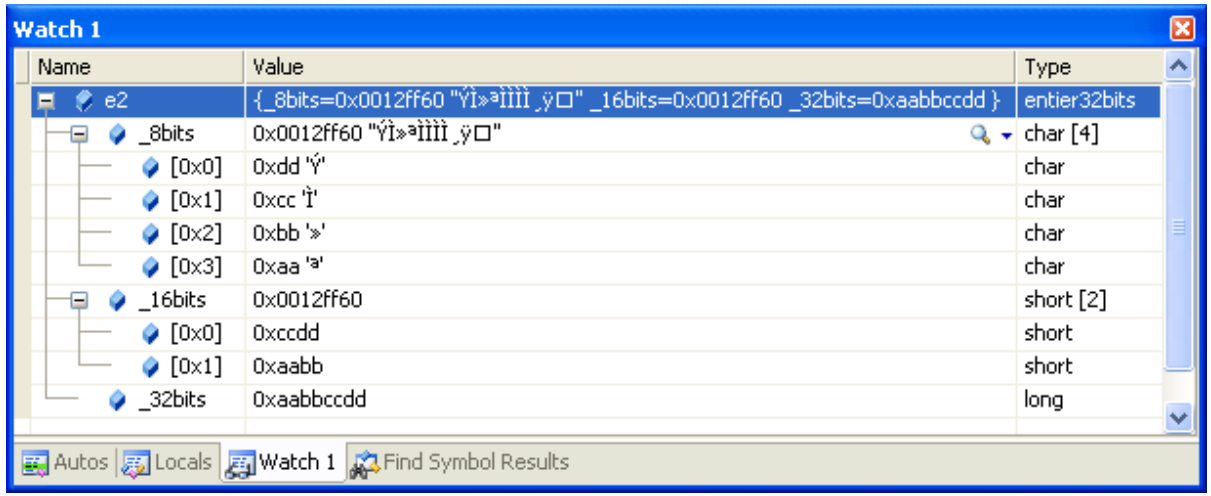
Des variables multiformes

- Décomposer un entier 32 bits en octets ou en mots de 16 bits

```
typedef union
{
    char _8bits[4];
    short _16bits[2];
    long _32bits;
} ENTIER32BITS;

int main()
{

    ENTIER32BITS e2;
    scanf("%lx", &e2._32bits);
    printf("%lx = { %x %x %x %x }\n",
        e2._32bits,
        e2._8bits[0], e2._8bits[1],
        e2._8bits[2], e2._8bits[3]);
}
```



Les unions

Application – un type pouvant contenir 2 formes de valeurs

```
typedef enum { cartesien, polaire } REPRESENTATION_COMPLEXE;

typedef struct
{
    double module;
    double argument;
} COMPLEXE_POLAIRE;

typedef struct
{
    double x, y;
} COMPLEXE_CARTESIEN;

typedef struct
{
    REPRESENTATION_COMPLEXE representation;
    union
    {
        COMPLEXE_CARTESIEN cartesien;
        COMPLEXE_POLAIRE polaire;
    } valeur;
} COMPLEXE;
```

Les champs de bits

Accéder facilement aux bits d'un entier

- Application : interpréter le contenu d'un registre
  - Exemple : commande d'un moteur

```
short registre_commande_moteur;
```

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
fonction	vitesse du moteur (0-255)								X	X	X	Sens	X	X	On	X

- Comment accéder facilement aux différents bits ?
  - Solution connue : utiliser les masques et les décalages de bit
    - Allumer le moteur :

```
registre_commande_moteur |= 1 << 1;
```

    - Inverser le sens :

```
registre_commande_moteur ^= 1 << 4;
```

    - Inconvénient : une certaine lourdeur d'écriture.
  - Autre solution : les champs de bits

Les champs de bits

Déclaration

bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
fonction	vitesse du moteur (0-255)								X	X	X	Sens	X	X	On	X

- Déclaration

```
typedef struct
{
    int :1; // inutilisé
    int on:1;
    int:2; // inutilisés
    int sens:1;
    int:3; //inutilisés
    int vitesse:8;
} REGISTRE_COMMANDE_MOTEUR;
```

Les champs de bits

Combinaison avec les unions

- Permet d'accéder
  - Soit bit par bit.
  - Soit globalement.
- Exemple

```
typedef union
{
    struct
    {
        int :1; // inutilisé
        int on:1;
        int:2; // inutilisés
        int sens:1;
        int:3; //inutilisés
        int vitesse:8;
    } commande_moteur;
    short registre;
}
REGISTRE_COMMANDE_MOTEUR;
```

Les champs de bits

Utilisation

- Assignation des champs de bits :

```
REGISTRE_COMMANDE_MOTEUR contenu;
contenu.registre = 0;
contenu.commande_moteur.vitesse = 100;
contenu.commande_moteur.sens = 0;
contenu.commande_moteur.on = 1;
```

Les champs de bits

Utilisation

- Au final, on souhaite écrire cet entier dans un registre.
- Comment écrire un champs de bit dans un registre ?
  - Adresse de destination absolue connue. Ex : 0x0020
  - Par transtypage :

```
*(int *)0x0020 = *(int *)&contenu;
```

  - Par accès à la valeur globale:

```
*(int *)0x0020 = contenu.registre;
```

Les champs de bits
Utilisation

- Accès direct au registre

```
#define MOTEUR (*(REGISTRE_COMMANDE_MOTEUR *)0x0020)

int main()
{
   // Initialiser le registre à 0
   MOTEUR.registre = 0;
   // Fixer la vitesse
   MOTEUR.commande_moteur.vitesse = 6;
   // Choisir le sens
   MOTEUR.commande_moteur.sens = 0;
   // Allumer le moteur
   MOTEUR.commande_moteur.on = 1;
   return 0;
}
```

Les champs de bits

Code généré

```
    // Initialiser le registre à 0
    MOTEUR.registre = 0;
0041144E  xor         eax,eax
00411450  xor         ecx,ecx
00411452  mov         word ptr [eax+20h],cx
    // Fixer la vitesse
    MOTEUR.commande_moteur.vitesse = 6;
00411456  xor         eax,eax
00411458  mov         ecx,dword ptr [eax+20h]
0041145B  and         ecx,0FFFF00FFh
00411461  or          ecx,600h
00411467  xor         edx,edx
00411469  mov         dword ptr [edx+20h],ecx
    // Choisir le sens
    MOTEUR.commande_moteur.sens = 0;
0041146C  xor         eax,eax
0041146E  mov         ecx,dword ptr [eax+20h]
00411471  and         ecx,0FFFFFFFEh
00411474  xor         edx,edx
00411476  mov         dword ptr [edx+20h],ecx
    // Allumer le moteur
    MOTEUR.commande_moteur.on = 1;
00411479  xor         eax,eax
0041147B  mov         ecx,dword ptr [eax+20h]
0041147E  or          ecx,2
00411481  xor         edx,edx
00411483  mov         dword ptr [edx+20h],ecx
          }
```

Qu'avons-nous appris ?

- Les types énumérés.
- Les unions.
- Les champs de bits.

Vos questions


Exercices
Types de données
avancés