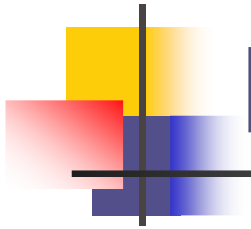
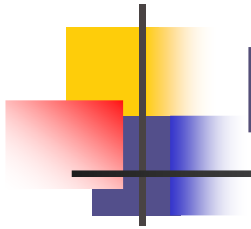


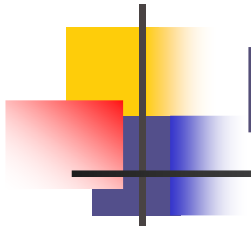
Les structures

- Permettent de grouper des données
- Exemples :
  - Données concernant l'identité d'une personne (nom, prénom, âge, adresse, etc)
  - Date : jour, mois, année
  - Coordonnées d'un point : x, y



Déclaration d'une structure

```
struct nom_structure {
  type1 membre1;
  type2 membre2;
} variable1, variable2,... ;
```



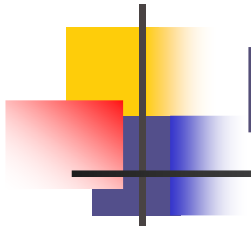
Exemple 1

- **Définition :**

```
struct coord {
  int x ;
  int y ;
};
```

- **Déclaration de variables :**

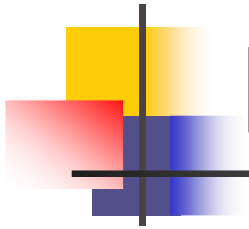
```
struct coord point1, point2 ;
```



Exemple 2

- Définition et déclaration :

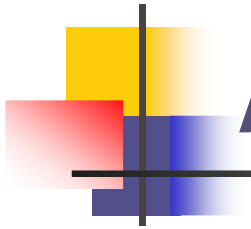
```
struct coord {
  int x ;
  int y ;
} point1, point2;
```



Exemple 3

```
struct heure {
  int heures ;
  int minutes ;
  int secondes ;
} heure_naissance = {14, 23, 5} ;
```

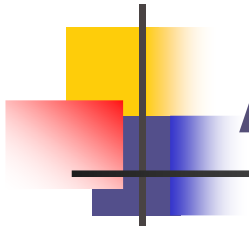
Avec initialisation statique



Accès aux membres

- L'opérateur .
- *Exemple :*

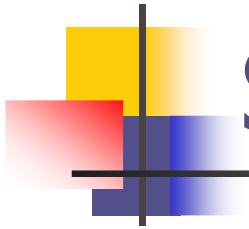
```
point1.x = 10 ;
point1.y = 50 ;

printf("Point %d %d\n",
                point1.x, point1.y);
```



Argument de fonctions

- Une structure est passée par valeur :

```
struct coord {
  int x ;
  int y ;
} ;
void printPoint (struct coord point)
{
  printf ("x : %d, y : %d\n",
                  point.x, point.y) ;
}
```

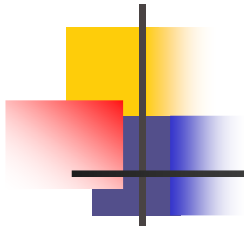


Structures imbriquées

```
struct rectangle {
  struct coord hautgauche ;
  struct coord basdroite ;
} boite;
```

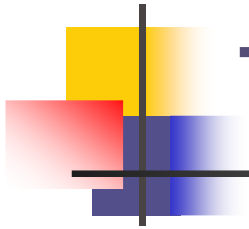
- Accès aux membres:

```
boite.hautgauche.x = 0 ;
boite.hautgauche.y = 10 ;
boite.basdroite.x = 100 ;
boite.basdroite.y = 200 ;
```



Structures imbriquées

```
struct typeDate {
   struct typeHeure {
       int heures ;
       int minutes ;
   } heure ;
   struct typeJour {
       int noJour ;
       int mois ;
    } jour ;
} date ;
date.heure.heures = 8 ;
date.heure.minutes = 30 ;
date.jour.noJour = 23 ;
date.jour.mois = 6 ;
```



Tableaux dans une structure

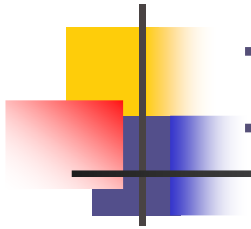
```
struct data {
  int x[3] ;
  char y[7] ;
} var1 = {1,2,3, "PARTEZ"}, var2 ;
```

- Affectation :

```
var2 = var1 ;
strcpy (var2.y, "STOP");
Valeur de var2.y[0] ?
```

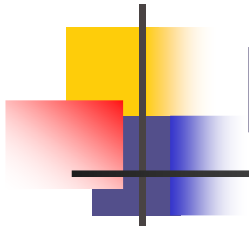
Tableaux dans une structure 2

```
struct date {
  char jour[3];
  char mois[3];
  char annee[5];
}date_jour={"14","09","1515"} ;
```



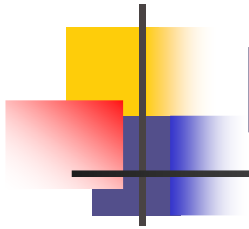
Initialisation statique

```
struct client {
   char societe[20] ;
   char contact[20] ;
};
struct vente {
   struct client acheteur;
   char article[20];
   float montant;
}mesventes={{"Office world", "Gérard Lambert"},
          "papier imprimante", 100.00};
```



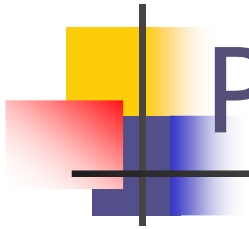
Pointeur sur une structure

```
struct coord {
  int x;
  int y;
} point;

struct coord *ptrPoint;

ptrPoint = &point;
```



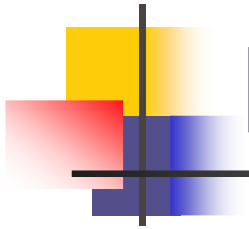
Pointeur et membre

```
(*ptrPoint).x = 10;
(*ptrPoint).y = 20;

ptrPoint->x = 10;
ptrPoint->y = 10;
```



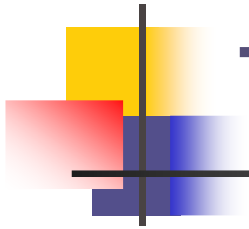
Pointeur membre de structure

```
struct typeDate {
  int jour ;
  int mois ;
  int annee ;
} date1={14, 7, 1789};
struct typeAnniversaire {
  char *nom ;
  struct typeDate *date ;
} anniversaire ;
anniversaire.nom = "Jean Dupond" ;
anniversaire.date = &date1 ;
```



Pointeur membre de structure

```
printf
   ("Nom %s date %d %d %d\n",
  anniversaire.nom,
  anniversaire.date->jour,
  anniversaire.date->mois,
  anniversaire.date->annee);
```



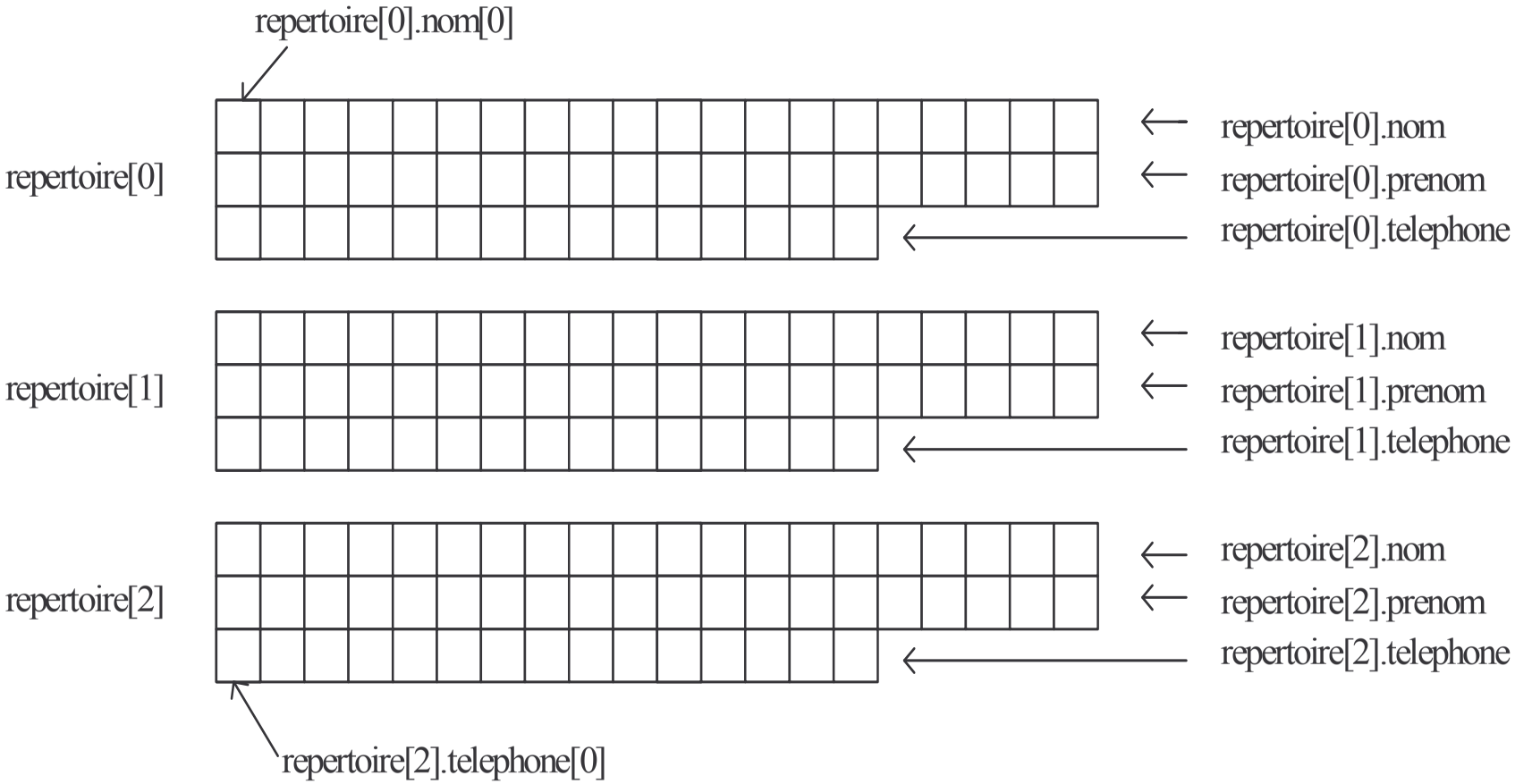
Tableaux de structures

```
struct typePersonne {
  char nom[20];
  char prenom[20];
  char telephone[15];
};
```

- Répertoire téléphonique :

```
struct typePersonne repertoire[100];
```

Tableaux de structures



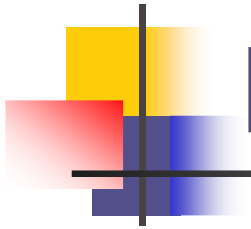
Pointeurs sur tableaux de structures

```
struct typePersonne *ptrPersonne ;
```

Initialisation :

```
ptrPersonne = repertoire ;
```

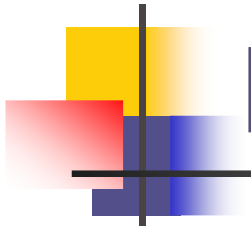
```
ptrPersonne++;
```



Les unions

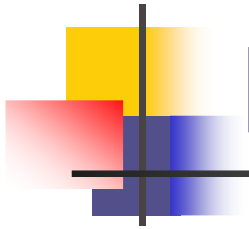
- Définition :

```
union nomUnion {
  type1 membre1;
  type2 membre2;
  ...
} variable ;
```



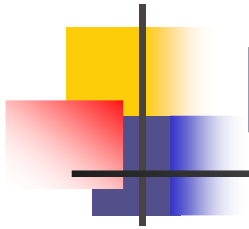
Les unions

- Un seul membre à la fois
- Taille : taille du plus grand membre
- Opérations : idem structure



Les unions - Exemple

```
union exemple {
  char car;
  int val;
} toto;
toto.car = 'a';
printf ("car : %c, val : %X\n",
          toto.car, toto.val);
toto.val = 0;
printf ("car : %c, val : %X\n",
          toto.car, toto.val);
```



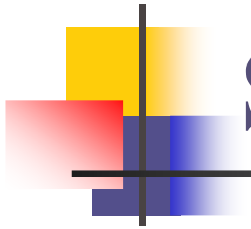
Les unions – Exemple (suite)

Résultat :

```
car : a, val : CCCCCC61
car :  , val : 0
```

ligne 1 : un seul caractère à été initialisé dans l'entier.

ligne 2 : le caractère n'est pas un caractère imprimable.



Structure et union

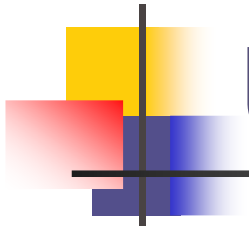
- Une union membre d'une structure

```
struct typeDonnee {

  int type ;
  union part {
     char car;
     int val;
  } donnee;
};
```

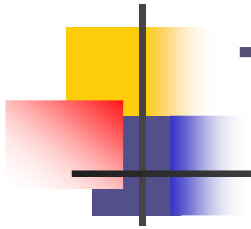
Utilisation d'une union

```
void printDonnee (struct typeDonnee d) {
  if (d.type == 0)
     printf ("Caractere : %c\n",
                        d.donnee.car);
  else
     printf ("Entier : %d\n",
                      d.donnee.val);
}
```

Utilisation d'une union (2)



```
void main () {
    struct typeDonnee d1 = {0, 'a'};
    struct typeDonnee d2 = {1, 100};

     printDonnee (d1);
     printDonnee (d2);
}
```

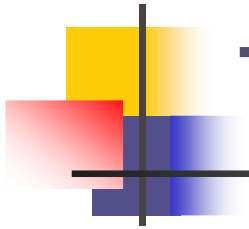


Typedef

Permet de définir un nouveau type.

```
typedef type nomdutype;
```

```
typedef int entier ;
entier valeur ;
```



Typedef et les structures

```
typedef struct typeCoord {
    int x;
    int y;
  } coord;

coord point1;
```