



## Chap. 9 Types utilisateurs (suite et fin)

# I2181A Langage C : modularisation

Anthony Legrand

# Les structures récursives

# Structure récursive

3


- C doit connaître la taille d'un type à la compilation !

```
struct Noeud {  
    int valeur;  
    struct Noeud suivant;  
};
```

# Structure réursive

- C doit connaître la taille d'un type à la compilation !

```
struct Noeud {  
    int valeur;  
    struct Noeud suivant;  
};
```



erreur: taille inconnue  
→ récursivité interdite !

# Structure récursive

- ▶ C doit connaître la taille d'un type à la compilation !
  - utilisation d'un pointeur

```
struct Noeud {  
    int valeur;  
    struct Noeud * suivant;  
};
```



correct:  
taille connue

typedef

# typedef (1)

- ▶ définition d'un synonyme d'un type existant

```
typedef unsigned int size_t;
```

# typedef (1)

- ▶ définition d'un synonyme d'un type existant

```
typedef unsigned int size_t;
```



mot clé



# typedef (1)

- ▶ définition d'un synonyme d'un type existant

```
typedef unsigned int size_t;
```



type existant

# typedef (1)

- ▶ définition d'un synonyme d'un type existant

```
typedef unsigned int size_t
```



nouvel  
identificateur  
de type

# typedef (2)

- ▶ souvent utilisé pour renommer les types complexes t.q. structures et énumérations

```
struct Point {  
    int abscisse;  
    int ordonnee;  
};  
  
enum Couleur {NOIR, JAUNE, ROUGE};
```



```
struct Point p;  
enum Couleur c;
```

# typedef (2)

- ▶ souvent utilisé pour renommer les types complexes t.q. structures et énumérations

```
typedef struct Point {  
    int abscisse;  
    int ordonnee;  
} Point;
```

```
typedef enum Couleur {NOIR,JAUNE,ROUGE} Couleur;
```



```
Point p;  
Couleur c;
```

# typedef (2)

- ▶ souvent utilisé pour renommer les types complexes t.q. structures et énumérations

```
typedef struct {  
    int abscisse;  
    int ordonnee;  
} Point;
```

```
typedef enum {NOIR, JAUNE, ROUGE} Couleur;
```



```
Point p;  
Couleur c;
```

# Exemple 1

14

- ▶ définition classique d'un type booléen en C (non existant en ANSI-C) :

```
                                0      1  
                                ||      ||  
typedef enum {FALSE, TRUE} Boolean;  
  
...  
  
Boolean valid = TRUE;
```

# Exemple 2

15

- définition d'un type file (FIFO) d'entiers :

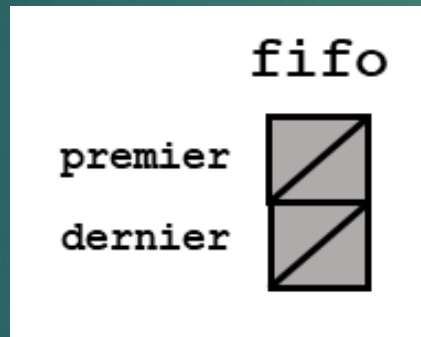
```
typedef struct Noeud {  
    int val;  
    struct Noeud* suivant;  
} Noeud;  
  
typedef struct {  
    Noeud* premier;  
    Noeud* dernier;  
} File;
```

# Example 2

16

## ► File vide:

```
File fifo = {NULL,NULL};
```





# Exemple 2

17

## ► File après ajouts :

```
enfiler(&fifo,1);  
enfiler(&fifo,2);  
enfiler(&fifo,3);
```

