Algorithmique et programmation

Pointeurs

R.Gosswiller

Le type Void

Le type Void

Définition

Void est un type de variable au même titre que Int ou Double. Il est parfois appellé le type 'incomplet'.

Principe

Void est utilisé dans les prototypes de fonction pour dénoter l'abscence d'argument d'entrée ou de renvoi.

Syntaxe

```
1 int fonction(int a);   //Fonction
2 void fonction(int a); //Procedure
3 int fonction(void);    //Sous-programme
4 int 'void'(int a);     //Pointeur
```

Attention

Le type void à de nombreuses restrictions qui lui sont attachées

1. Sizeof(void) ne renvoie pas toujours le même résultat selon les OS (souvent 0).
2. Les opérations arithmétiques +, -, *, / (et autres) ne sont pas utilisables avec void.
3. Les opérations de comparaison <, =, > (et autres) ne fonctionnent pas non plus.

Principe

Void est utilisé afin de noter l'existence de quelque chose sur lequel on ne dispose pas encore d'informations.
Notamment dans les pointeurs.

Pointeur générique

Void*

La syntaxe Void* permet de créer un pointeur générique.

Syntaxe

```
1 (void *)a; // a est un pointeur generique
2 void *b;   // correct egalement
```

Adressage

Comme il n'existe aucune information attachée à un pointeur générique, n'importe quelle adresse peut y être attachée

Exemples

```
1  int a;
2  double b;
3  void *p;
4  double *r;
5
6  p = &a; // Correct
7  p = &b; // Correct
8  r = p; // Correct
9
10 p = &a;
11 r = p; // risque de provoquer des erreurs!
```

Exemple d'utilisation de void*

Utilisation

```
1  int a;
2  int *p = &a;
3
4  printf("%p==%p\n", (void *)&a, (void *)p);
5  //ici void* permet d'afficher l'adresse de n'importe quel type
6
7  printf("%p\n", (void *)NULL);
8  //Affiche l'adresse invalide de l'OS (normalement 0)
```