Initiation à l'Algorithmique

Les sous Programmes

ENS KATEB

Introduction

Problématique n°1

Comment développer de grosses applications en utilisant que les structures de base?

Solution

1 - Diviser le gros PB en sous petits PB
   - Individuellement moins complexes,
   - Aussi indépendants les uns de les autres que possible

2 - Résoudre ces sous-problèmes

3 - Assembler les solutions

Exemple : Concevoir une voiture :

Sous PB : carrosserie, habitacle, moteur, transmission, système électrique, …

Problématique n°2

Comment éviter de résoudre plusieurs fois les mêmes problèmes ?

Solution

Utiliser une technique permettant la résolution d'un problème de manière qu'il soit possible de faire appel à cette technique sans avoir à la réécrire

Exemples:

En Math : sin(x) -> Calcul de sinus(x)

Introduction

- Une fonction : est un sous Prg qui prend des paramètres en entrée et qui retourne une valeur de résultat en sortie selon un type bien défini.
- Une procédure : est une fonction qui ne retourne rien en sortie et sans type.
- Déclaration :
  - **Fonction** Nom_Fonction (arguments) : type
    Début
      Var
        Déclarations;
      ……
      ……
      Retourner val;
    Fin;
  - **Procédure** Nom_Procédure (arguments)
    Début
      Var
        Déclarations;
      ……
      ……
    Fin;

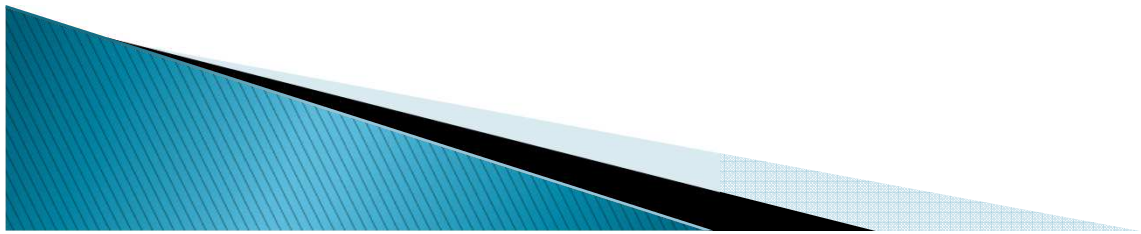
NB : En Algorithmique on a deux primitives : Fonction et Procédure

Introduction

En C++ :

- type Nom_Fonction (paramètres)
  {
  ……
  ……
  return val;
  }
- **void** Nom_Procédure (paramètres)
  {
  ……
  ……
  }

NB : En C++ on a une seule primitive : Fonction et pour une procédure on a la fonction vide (void)



Exemples

- Exemple 1 :

```
Fonction max(a, b : entier) : entier
Début
    Var m : entier;
    Si (a<b) alors
        m ← b;
    Sinon
        m ← a;
    Fsi;
    Retourner m ;
  Fin;
```

Cette fonction retourne le maximum entre deux nombres qui sont les arguments de notre fonction

Dans le programme principale :

```
Algo Ex01
  Var X,Y : entier;
  Début
    écrire (''Donner deux nbrs:'');
    lire (X,Y);
    écrire (''le maximum est:'',
                max(X,Y) );
  Fin.
```

C'est l'appel de la fonction :
Les données a et b sont remplacées par des valeurs
La fonction est exécutée jusqu'au premier return.
Le résultat **retourné par la fonction est la valeur de** l'expression du return.
Ce résultat (valeur) est affiché par la commande écrire.

Exemples

- Exemple 2 :

Écrire un s/prg qui permet d'indiquer si un nombre est multiple de 3 ou non, puis écrire un programme qui permet de vérifier les nombres multiples de 3 entre 1 et 100.

```
Fonction mul3 (A : entier) : booleen
Début
   si ((A mod 3)=0) alors
        retourner vrais;
   sinon
        retourner faux;
   finsi;
Fin;
Algo Ex02
Var i: entier;
Début
  pour(i de 1 à 100) faire
     si( mul3(i)) alors
         écrire (i , '' est multiple de 3\n'');
      sinon
         écrire (i , '' n'est pas multiple de 3\n'');
      Fsi;
   Fpour;
Fin.
```

```
bool mul3 (int A)
{
   if ((A%3)==0)
        return true;
   else
        return false;
}

int main()
{
  int i;
  for(i=1; i <=100; i++)
     if (mul3(i))
         cout << i << '' est multiple de 3\n'';
      else
         cout << i << '' n'est pas multiple de 3\n'';
   return 0;
}
```

Façons de déclarer les fonctions

Dans le mème fichier .cpp

```
float fexple (float x, int b, int c)
{
....
}
main()
{
  .....
  y = fexple (x, n, p) ;
  .....
}
```

1

```
main()
{
  float fexple (float, int, int) ;// prototype
  .....
  y = fexple (x, n, p) ;
  .....
}
float fexple (float x, int b, int c)
{
....
}
```

2

Dans deux fichiers : 1ere .cpp le 2eme .h

```
#include ''fonction.h''
main()
{
  .....
  y = fexple (x, n, p) ;
  .....
}
```

Où le fichier fonction.h contient le code de la fonction fexple

3

```
#include ''fonction.h''
main()
{
  .....
  y = fexple (x, n, p) ;
  .....
}
```

Utiliser deux fichiers : un .h et autre .cpp
Où le fichier fonction.h contient le prototype de la fonction fexple et le fichier fonction.cpp contient le code de la fonction fexple, au total on aura 03 fichiers.

4

Façons de déclarer les fonctions

- En Algorithmique: Deux méthodes :

En Début	En Dernier
Fonction nom_fct(arguments) : type Début Var Déclarations; …… Fin; Algo nom_algo Var Déclarations; Debut appel à nom_fct(……); …… Fin.	Algo nom_algo Var Déclarations; Prototype de la fonction fct; Debut appel à nom_fct(……); …… Fin. Fonction nom_fct(arguments) : type Début Var Déclarations; …… Fin;