Initiation à l'Algorithmique

Les sous Programmes (Suite)

ENS KATEB

Façons de déclarer les fonctions

- Exemple :

  Écrire un sous prg qui permet de renvoyer la valeur d'une fonction f(x) = $4X^3 + 2X^2 - 6X - 10$, puis l'algorithme qui affiche la valeur de cette fonction pour les nombre de 0 à 10

```
Algo exemple
Var   i : entier;
Fonction f(entier) : entier;//prototype
Début
   pour (i de 0 à 10) faire
     écrire (f(i), ''\n'');
   Fpour
Fin.
Fonction f(x : entier) : entier
Début
   retourner (4*pow(x,3) +
                2*pow(x,2) - 6*x - 10);
Fin;
```

OU

```
Fonction f(x : entier) : entier
Début
   return (4*pow(x,3) + 2*pow(x,2) -
                6*x - 10);
Fin;
Algo exemple
Var   i : entier;
Début
   pour (i de 0 à 10) faire
     écrire (f(i), ''\n'');
   Fpour
Fin.
```

Variables locales et variables globales

- Les variables locales :
  Une variable est locale lorsqu'elle est reconnu seulement dans le bloc où elle est déclarée
- Ex :

```
main()
{
  int n=10, p=20 ;
  if(n<10)
     int s = n+2;
  else
     for(int i=1; i<10;i++)
        cout << i+n << "\n" ;
}
```

La porté des variables n et p se propage jusqu'à la fin de la fonction main, pour s elle est limité dans le bloc if et pour i elle est reconnu seulement à l'intérieure de la boucle for

Variables locales et variables globales

- Les variables Globales :
  Une variable est globale lorsqu'elle est partagée par plusieurs fonctions y compris la fonction main
- Ex :

```
int n;
main()
{ n = 9;
  cin >> n;
}
int max(int a, int b)
{
  a = n;
}
float fct (float x)
{
  return x*n;
}
```

La variable n est utilisable dans toutes les fonctions main, max et fct quelque soit son emplacement

NB : il faut prendre en considération les changements de la valeur de n par les différentes fonctions.

Variables locales et variables globales

- En Algorithmique :

```
Algo ex
 Var
   A : entier; //A est globale
 Début
   ……
   ……
 Fin.
```

```
Algo ex
  Début
    Var
      A : entier; //A est locale
    ……
    ……
   Fin.
```

Passage des arguments

- Passage par valeur :

ex :

```
Algo ex
Var  X, Y : entier;
procédure permut(entier, entier);
Début
    X ← 3;
    Y ← 9;
    permut (X,Y);
    écrire (X,Y);
 Fin.
 procédure permut (A, B : entier)
    Début
       var t : entier;
       t ← A;
       A ← B;
       B ← t;
     Fin;
```

Normalement le programme affiche après l'exécution 9 3
Mais dans ce cas il va afficher 3 9
C'est la cause de passage des arguments par valeur :
les valeurs des arguments en sortie de la fonction ou procédure auront les mêmes valeurs qu'en entrée même s'ils ont subit des changements à l'intérieur de la fonction ou la procédure.

Passage des arguments

- Passage par référence :

ex :

```
Algo ex
Var  X, Y : entier;
procédure permut(var  entier, entier);
Début
    X ← 3;
    Y ← 9;
    permut (X,Y);
    écrire (X,Y);
 Fin.
 procédure permut (var A, B : entier)
    var t : entier;
    Début
       t ← A;
       A ← B;
       B ← t;
     Fin;
```

Ici on a indiqué que les arguments de la procédure permut A et B sont variables on parle donc de passage par référence, les arguments prendront en sortie des valeurs selon les changement effectués à l'intérieur de la procédure et non pas les valeurs en entrée.

Passage des arguments

En C++ :

Passage par valeur :

```
void permut (int A, int B)
  {
    int t = A;
    A = B;
    B = t;
   }
```

Passage par référence

```
 void permut (int &A, int &B)
    {
      int t = A;
      A = B;
      B = t;
     }
```

On ajoute dans l'entête devant les arguments le symbole **&**

Exemple

- Écrire un programme en C++ qui permet de calculer le coefficient binomial

$$C_n^k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Avec n ≥ 0 et k ≥ 0 et k ≤ n

Diviser le problème en sous pb:

1: Lecture de n et k où la lecture doit renvoyer : n ≥ 0 et k ≥ 0 et k ≤ n

2: Calcule du factoriel d'un nombre n!

3: Calcule de la valeur A/B*C

Solution

```
#include <iostream>
using namespace std;

void lect(int&, int&);
long fact(int);
float div(int, int, int);

int main()
{
  int n,k;
  float c;
  lect(n,k);
  c = div(fact(n), fact(k), fact(n-k));
  cout << "\nCoeffecient Binomial = " << c;
}

float div(long A, long B, long C)
{
    return A/(B*C);
}
```

```
void lect(int &N, int &P)
{
 do
  {
    cout << "\nEntrer le N:";
    cin >> N;
    cout << "Entrer le P:";
    cin >> P;
    if((N<0)||(P<0))cout << "Vous aves taper
                          un nombre négatif!";
    if(N<P)cout <<"Vous aves taper N < P!!";
  }
 while ((N<P)||(N<0)||(P<0));
}
long fact(int N)
{
   long F=1;
   while(N>1)
     {
       F*=N;  N--;
     }
   return F;
}
```