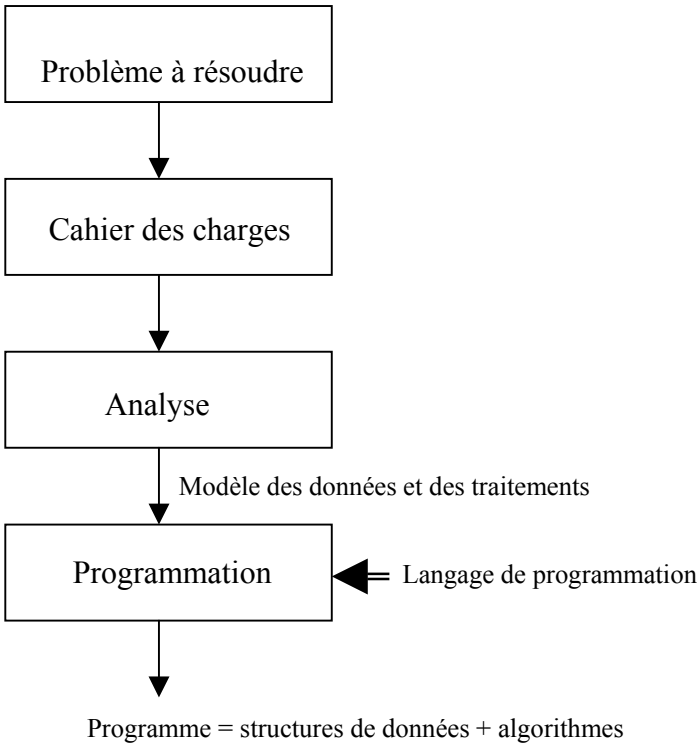
2.3- Conception d'un programme



2.4- De l'algorithme au programme

2.4.1- Considérations générales

Un *algorithme* est une méthode systématique pour résoudre un problème donné en un temps fini. Une recette de cuisine précise est un algorithme ; la méthode de calcul du PGCD de deux entiers est un algorithme ; … .

En général, il existe plusieurs algorithmes possibles pour résoudre un même problèmes. Le choix d'un algorithme dépend de plusieurs critères, notamment :

- quantité et nature des données à traiter
- place mémoire et temps de calcul nécessaires
- fiabilité du résultat, sensibilité aux erreurs arithmétiques, …
- possibilité de parallélisme
- simplicité, clarté, facilité de réalisation
- type d'utilisation prévue (occasionnelle, industrielle, …)

Un algorithme peut être spécifié de façon plus ou moins formelle, mais néanmoins toujours totalement rigoureuse sur le plan sémantique :

- en langage naturel : description informelle la plus générale
- en « pseudo-langage » algorithmique : pour une description à un niveau d'abstraction intermédiaire. Un « pseudo-langage » algorithmique est un mélange de constructions classiques présentes dans tous les langages de programmation (boucles, conditions, …) et de langage naturel. L'intérêt de ce niveau de

description intermédiaire entre le langage naturel et le programme est de laisser la liberté de ne pas entrer trop vite dans les détails, de garder une vue globale de l'algorithme aux débuts de sa conception.

- en langage de programmation : description formelle ultime. L'algorithme est le programme.

Il est conseillé de passer par ces différents niveaux de langage de description d'algorithme, en raffinant successivement jusqu'au niveau de détails voulu. Nous allons illustrer la démarche sur un exemple simple.

2.4.2- Exemple

- Cahier des charges : écrire un programme C/C++ réalisant une multiplication de deux entiers naturels par l'algorithme de « multiplication à la russe »
- Algorithme en langage naturel (on suppose d'abord une réalisation de la multiplication à la main sur papier)

> « Ecrire le multiplicateur et le multiplicande côte à côte. Former une colonne sous chaque opérande en répétant la règle suivante jusqu'à ce que le nombre sous le multiplicateur soit égal à 1 :
>
> > diviser par 2 par une division entière le dernier nombre de la colonne multiplicateur et doubler le dernier nombre de la colonne multiplicande.
>
> Rayer tous les nombres de la colonne multiplicande correspondant à un nombre pair sous le multiplicateur. Additionner les nombres restant sous le multiplicande. Le résultat de l'addition est le résultat cherché. »

Exemple : soit à calculer 19 x 45

45	19
22	~~38~~
11	76
5	152
2	~~304~~
1	608
	855

A noter que cet algorithme possède des propriétés qui le rende facile à réaliser par un dispositif électronique . Sous diverses variantes, il est utilisé dans des circuits de calcul de nombreux ordinateurs.

- Algorithme en pseudo-langage version 1

```
Soient A et B les deux entiers naturels à multiplier ;
Soit R le résultat à trouver ;
Si A est impair alors initialiser R à la valeur de B, sinon initialiser R à 0 FinSi ;
Tant que A > 1 faire
        diviser A par 2 par une division entière ;
        multiplier B par 2 ;
        si A est impair alors ajouter B à R ;
FinTantQue.
```

Exemple : soit à calculer 19 x 45. Les valeurs successives des variables A, B et R aux différentes étapes de l'algorithme sont :

A	B	R
45	19	19
22	38	19
11	76	95
5	152	247
2	304	247
1	608	855

Cet algorithme est correct mais on peut améliorer ses performances en choisissant le plus petit opérande pour la colonne de gauche (minimise le nombre d'opérations). D'où la version affinée suivante.

- Algorithme en pseudo-langage version 2

```
Procédure MultALaRusse (données A, B : entiers naturels ; résultat R : entier naturel)
        // Calcule AxB par multiplication à la russe et met le résultat dans R
        Si A > B alors Echanger(A, B) Finsi ;
        Si Impair(A) alors R← B sinon R ← 0 FinSi ;
        Tant que A > 1 faire
                A ← A div 2 ;
                B ← B + B ;
                Si Impair(A) alors R ← R + B finsi ;
        FinTantQue ;
FinProcédure.

Programme d'essai
        Demander à l'utilisateur de fournir 2 entiers naturels ;
        Soient i et j ces deux entiers ;
        Soit k le résultat cherché ;
        MultALaRusse(i, j, k) ;
        Afficher le résultat k ;
FinProgramme.
```

Ce niveau de description est une forme finale possible.

- Algorithme en C++ (i.e. programme)

```
/*             Multiplication a la russe              */
/*       (sous-programme + programme d'essai)         */
/*                    auteur - date                   */

/*****************************************************/
/* Declarations globales                             */
/*****************************************************/

#include <iostream.h>
typedef unsigned int  TypEntNat ;
void MultALaRusse ( TypEntNat a, TypEntNat b, TypEntNat & r ) ;
TypEntNat Mult_A_La_Russe ( TypEntNat a, TypEntNat b ) ;

/*****************************************************/
/* Programme principal d'essai                       */
/*****************************************************/
```

```
int main ()
{
	TypEntNat i, j, k ;

	cout << "Entrez deux entiers naturels : "  ;
	cin >> i >> j ;

	MultALaRusse(i, j, k);
	cout <<" Avec la procedure : " <<i <<'*' <<j <<"= " <<k <<endl;

	k = Mult_A_La_Russe(i, j);
	cout <<" Avec la fonction : " <<i <<'*' <<j <<"= " <<k <<endl;

	return 0 ;

} // fin main

/*****************************************************/
/* Procedure de multiplication a la russe            */
/* Parametres d'entrée : deux entiers naturels a et b */
/* Parametres de sortie : le produit de a et b       */
/*****************************************************/

void MultALaRusse (TypEntNat a, TypEntNat b, TypEntNat & r )
{
	if (a>b) { r=b; b=a; a=r; }
	if (a%2) r=b; else r=0;
	while (a>1)
	{
		a = a/2 ;
		b = b+b ;
		if (a%2) r=r+b ;
	}
} // fin procedure MultALaRusse

/*****************************************************/
/* Fonction de multiplication a la russe             */
/* Parametres : deux entiers naturels a et b         */
/* Valeur de la fonction : le produit de a et b      */
/*****************************************************/

TypEntNat Mult_A_La_Russe (TypEntNat a, EnTyptNat b)
{
	TypEntNat r ;

	if (a>b) { r=b; b=a; a=r; }
	if (a%2) r=b; else r=0;
	while (a>1)
	{
		a = a/2 ;
		b = b+b ;
		if (a%2) r=r+b ;
	}

	return r ;

} // fin fonction Mult_A_La_Russe
```