Algorithmes et programmes

Lorsqu'on a une tâche complexe à effectuer, il est nécessaire de la décomposer en étapes ou en opérations plus simples (et connues). La description de cette succession d'opérations qui permet d'aboutir au résultat recherché s'appelle un algorithme.

Définition de l'Association française de normalisation : "Ensembles des règles opératoires et de procédés définis en vue d'obtenir un résultat déterminé au moyen d'un nombre fini d'opérations".

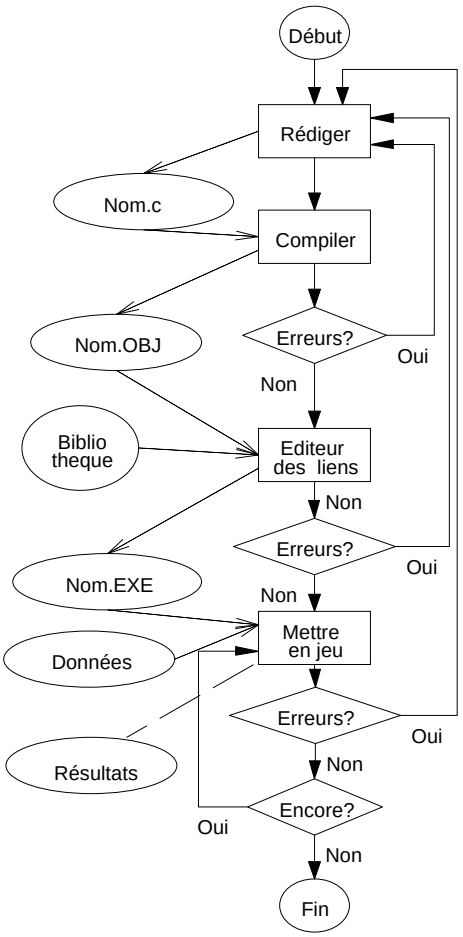


Fig. 1.1 Organigramme

Langages

1. Langages graphiques

1a. Organigrammes

Exemple: Algorithme de construction et mise en jeu d'un programme. (Fig.1.1)

1b. Arbres algorithmiques

Exemple Fig.1.2

2. Langages algorithmiques – C

Les caractères utilisés
Lettres - A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,
a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z, _
Chiffres – 0,1,2,3,4,5,6,7,8,9
Caractères spéciaux – . , ; : ' " [ ] { } ( ) - + = * / % < > & ! ?
Des mots réservés.
Avec les caractères on forme des identificateurs (noms) et des instructions (phrases).Chaque nom d'un objet est constitué d'un nombre arbitraire de lettres et chiffres, mais le premier caractère doit être une lettre.
Exemples: interet Compte A25 BASE_3_G un_homme_triste
Chaque objet a un ensemble de caractéristiques, qui sont:

1. Un nom (identificateur)
2. Un type, qui définit quelles sont les valeurs que l'objet peut contenir, leur intervalle et (qui est le plus important) les opérations qui sont permises pour la modification de ces valeurs.
3. Une valeur
4. Une adresse dans la mémoire.
5. Un genre

Les objets peuvent être des constantes ou des variables. La valeur d'une constante ne peut pas être changée au cours d'exécution du programme.

Le C a 3 types prédéfinis: entier (int, byte, short, long), réel (float, double), caractère (char).

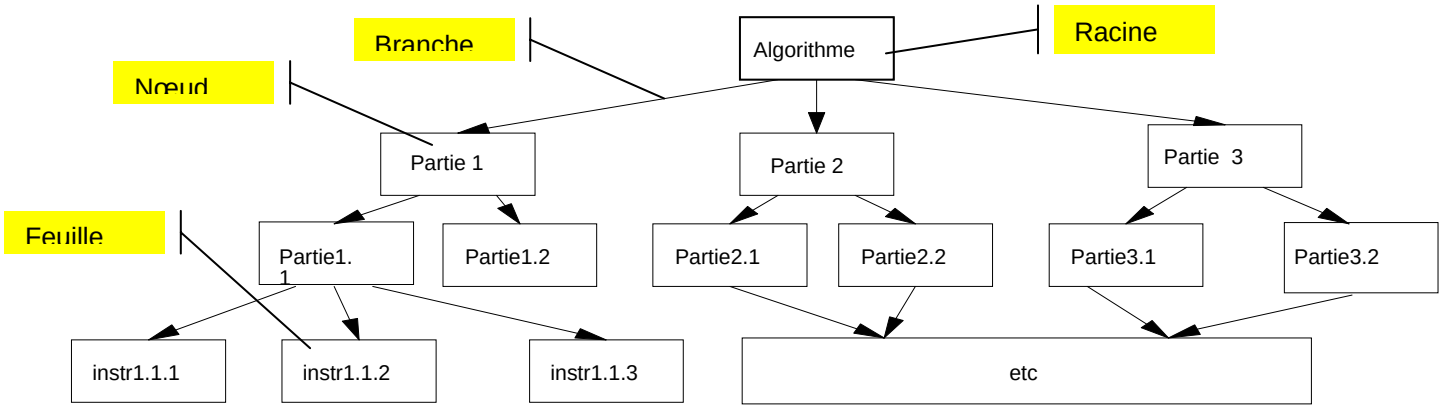


Fig 2.2. Arbre programmatique

Structure générale d'un programme: (le tableau ci-dessous)

<table>
<tr><td>Déclarations globales et Instructions de préprocesseur</td><td></td><td colspan="2">#include
#define
etc.</td></tr>
<tr><td rowspan="5">Fonctions (au moins une fonction nommée “main”).</td><td>Titre de la fonction</td><td colspan="2">Le type et le nom de la fonction et les types et les noms de ses paramètres
Type nom (type p1, type p2 , ….)</td></tr>
<tr><td rowspan="4">Corps de la fonction représentant un block</td><td rowspan="4">Chaque block peut contenir d'autres blocks</td><td>{</td></tr>
<tr><td>Déclarations des variables locales</td></tr>
<tr><td>Instructions exécutives (qui peuvent contenir des autres blocks)</td></tr>
<tr><td>}</td></tr>
</table>

A n'importe quelle place dans le programme on peut écrire des commentaires entre les pair de caractères /* et */. Le commentaire est ignoré par le compilateur.
Exemple :

```
#include <stdio.h>    /* Inclure les
   déclarations des variables et fonctions
   d'entrée/sortie */
void main(void) /* le titre de la fonction */
  {  /* Debut du block */
   printf ("Bonjour tout le monde\n");
                       /*Instruction */
  } /*Fin du block*/
```

Déclaration des objets d'un type prédéfini

Le type entier(int, short, long, unsigned, byte)

Une constante manifestée est écrite par sa valeur décimale, octale ou hexadécimale. Elle ne peut contenir que des chiffres appropriés avec ou sans un signe moins.
Exemples: 1000 -100 -2 5600 – constantes décimales
0234 0123 077 – constantes octales.
0x34A 0x10 0x123 – constantes hexadécimales.
On peut déclarer un nom d'une constante et utiliser ce nom au lieu de la valeur. Le nom doit être déclaré par l'instruction du préprocesseur #define
Exemples:

```
#define N 10
#define M 100
#define NOMBRE  150
```

Si nous voulons forcer l'ordinateur à utiliser un type de notre choix, nous pouvons employer les suffixes suivants:

suffixe	***type***	***Exemple***
u ou U	**unsigned (int ou long)**	**550u**
l ou L	**long**	**123456789L**
ul ou UL	**unsigned long**	**12092UL**

Les variables sont déclarées par leur nom et le nom de leur type. La syntaxe de l'instruction de déclaration est:
nom d'un type nom1, nom2,...,nomN ;

Exemples:

```
int un_entier;
short   i,j,k;
long grand_nombre ;
```

Les entiers sont représentés dans la mémoire comme des nombres binaires. L'intervalle des entiers dépend de l'organisation de la mémoire.

Type réel (float, double)

Le type par défaut et **double** qui comporte 64 bits (32 pour **float**).
La présentation des valeurs réels est avec une virgule flottante (floating point).
Les constantes manifestées ont toujours un point décimal ou une exposante.
Exemples:

```
 1.5   -7.25   0.675
  1.5E20  = 1.5.10^20
   1E-3  = 0.001
```

On déclare les constantes déclarées et les variables comme celles du type entier.
Exemples

```
#define  PI  3.14169265
#define EPS  1E-3
  float      a,b,c;
 double x,y,z ;
```

Dans la mémoire les réels sont présentés comme des nombres binaires avec une virgule flottante.

Le type logique

Les objets de ce type s'utilisent quand on doit déterminer l'accomplissement d'une condition. Il y a deux valeurs différentes **Vrai (True)** et **Faux (False). Le type logique n'existe pas en C**. Il est présenté par le type **int.** La valeur 0 présents la valeur Faux et chacune des autres valeurs, mais notamment -1 – la valeur Vrai.

Le type caractère (char)

Les constantes - caractères sont encadrées par des apostrophes. Par exemple **'a' 'A' '%' '3'.** On peut les dénoter par la séquence **\d$_1$d$_2$d$_3$** où d_i sont des chiffres octaux et présentent le code de caractère en système octale. Par exemple **'\062' '\120'.** On peut les écrire aussi par **\xd$_1$d$_2$** où d_i sont des chiffres hexadécimaux, par exemple: **'\x41' `\xA2'**

Les constantes–chaînes de caractères sont encadrées par des guillemets. Si la chaîne de caractères contient une guillemet ou quelque caractère spécial on utilise le tableau des séquences d'échappements. Une séquence d'échappement est un couple de symboles dont le premier est le *signe d'échappement* '\'. Au moment de la compilation, chaque séquence d'échappement est traduite en un caractère de contrôle dans le code de la machine. Comme les séquences d'échappement sont identiques sur toutes les machines, elles nous permettent d'écrire des programmes portables, c.-à-d.: des programmes qui ont le même effet sur toutes les machines, indépendamment du code de caractères utilisé.

Séquences d'échappement

\a	sonnerie	****	barre oblique
\b	curseur arrière	**\?**	point d'interrogation
\t	tabulation	**\'**	apostrophe
\n	nouvelle ligne	**\"**	guillemets
\r	retour au début de ligne	**\f**	saut de page (imprimante)
\0	NUL	**\v**	tabulateur vertical

La constante **'\0'** qui a la valeur numérique zéro (ne pas à confondre avec le caractère '0' !!) a une signification spéciale dans le traitement et la mémorisation des chaînes de caractères: En C le caractère '\0' définit **la fin d'une chaîne** de caractères.

Exemples:

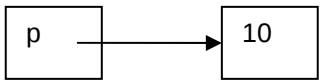
```
#define  C1  'A'
#define  CHAINE1  "TINTIN"
#define  CHAINE2  "c\'est un \"exemple\"\?"
la valeur de la dernière constante est c'est un "exemple"?
```

En effet le type **char** est un entier d'une longueur un octet avec ou sans signe en fonction de la réalisation. Si on veux de spécifier on utilise les types **signed char** ou **unsigned char.**

Les pointeurs

Le pointeur est un spécial type de données dont la valeur est l'adresse d'une location de la mémoire qui contient la valeur d'un certain type – le type du pointeur. On dit que le pointeur désigne la valeur. Mais la valeur du pointeur lui-même est

un nombre entier qui est l'adresse de la location. Comme un entier le pointeur peut subir certaines opérations, qui constituent l'arithmétique des pointeurs.
Le pointeur est et la location dont l'adresse et la valeur sont présentés de la façon suivante :



La déclaration du pointeur est faite dans les instruction de déclaration de variables du type sur lequel est le pointeur, mais on ajoute avant le nom du pointeur le caractère * :

```
int *p ; /* c'est un pointeur sur int */
double x, *pd  /*une variable de type double et un pointeur sur double*/
```

Les pointeurs ont une valeur (et constante) spéciale NULL qui dénote que le pointeur ne désigne rien. Ce n'est pas la constante 0 car la mémoire a une cellule dont l'adresse est zéro.