6. Les constantes

Il arrive parfois que l'on ait besoin d'utiliser une variable dont on voudrait qu'elle garde la même valeur pendant toute la durée du programme. C'est-à-dire qu'une fois déclarée, vous voudriez que votre variable conserve sa valeur et que personne n'ait le droit de changer ce qu'elle contient.

Ces variables particulières sont appelées **constantes**, justement parce que leur valeur reste constante.

Pour déclarer une constante, c'est en fait très simple : il faut utiliser le mot `const` juste devant le type quand vous déclarez votre variable. Par ailleurs, il faut obligatoirement lui donner une valeur au moment de sa déclaration. Après, il sera trop tard : vous ne pourrez plus changer la valeur de la constante.

L'utilisation des constantes est primordial, elle permet de modifier la dimension d'un problème et de son traitement avec un minimum de manipulations.
La déclaration des constantes se fait comme suit :

6.1.En algorithmique

Constantes
réel Pi = 3.1415927
entier NbrCoins = 4

6.2.En langage C

```
const float Pi=3.14159;
const int NbrCoins=4;
```

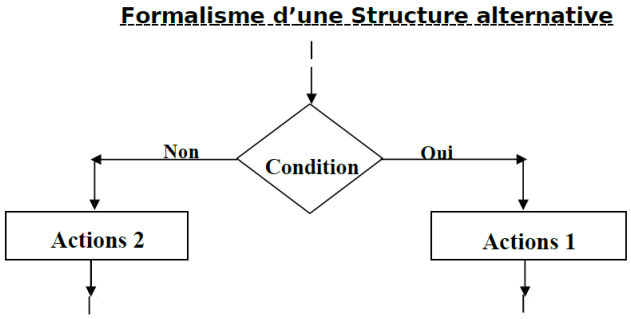
Exemple

```
#include <stdio.h>
#include <math.h>
main()
{
const float Pi=3.14159;
float R,S,C ;
printf  (" Entrez le rayon du cercle: ");
scanf  ("%f",&R);
S=Pi* (pow (R,2));
C=2*Pi*R;
printf ("La surface de cette cercle = %f \n",S);
printf ("La circonference du cercle = %f\n",C);
}
```

7. Les structures conditionnelles (alternatives)

Il est souvent nécessaire lorsque l'on écrit un programme de distinguer entre plusieurs cas conditionnant l'exécution de telles ou telles instructions. Pour ce faire, on utilise une structure alternative (Conditionnelle) : **Si on est dans tel cas on fait cela sinon on fait ceci.**

Le formalisme de cette structure dans un organigramme sera comme suit :



Dans les structures conditionnelles on se base sur ce qu'on appelle ***prédicat*** ou ***condition***. Ce dernier est un énoncé ou proposition qui peut être **vrai** ou **faux** selon ce qu'on est entrain de parler. Dans l'établissement des conditions on utilise des opérateurs dit de comparaison, à retenir :

Symbole en Algo	Symbole en C	Rôle	Formalisme
<	<	Strictement inférieur à ...	x < y (x est inférieur à y)
<=	<=	Inférieur ou égal à ...	x <= y (x est inférieur ou égal à y)
>	>	Strictement supérieur à ...	x > y (x est supérieur à y)
>=	>=	Supérieur ou égal à ...	x >= y (x est supérieur ou égal à y)
=	==	Egal à ...	x == y (x est égal à y)
≠	!=	Différent de ... (non égal à ...)	x != y (x est différent de y)

Comme on utilise des opérateurs dit logiques, à retenir également :

Symbole en Algo	Symbole en C	Rôle	Formalisme
Ou	||	OU logique Vérifie qu'une des conditions est réalisée	((condition1) || (condition2))
Et	**&&**	ET logique Vérifie que toutes les conditions sont réalisées	((condition1) && (condition2))
Non	**!**	NON logique Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	!(condition)

7.1. Types de structures de contrôle conditionnelles

Il existe trois formes d'instructions conditionnelles:

- ➢ Forme simple
- ➢ Forme alternative
- ➢ Forme à choix

7.1.1. ***Forme Simple***

Une structure de contrôle conditionnelle est dite à forme simple (réduite) lorsque le traitement dépend d'une condition. Si la condition est évaluée à « vrai », le traitement est exécuté.

Vocabulaire et syntaxe:

Algorithme	***Langage C***
Si condition **Alors** Instruction 1 Instruction 2 …. Instruction N **FinSi**	**if** (*expression*) */*obligatoirement entre parenthèses */* ***{*** *Instructions* ***}***

7.1.2. ***Forme alternative:***

Une structure de contrôle conditionnelle est dite à forme alternative lorsque le traitement dépend d'une condition à deux états: Si la condition est évaluée à « vrai », le premier traitement est exécuté, si la condition est évaluée à « faux », le second traitement est exécuté.

Vocabulaire et syntaxe:

Algorithme	***Langage C***
Si condition **Alors** Instruction 1.1 Instruction 2.2 …. Instruction N.1 **Sinon** Instruction 1.2 Instruction 2.2 …. Instruction M.2 **FinSi**	**if** (*expression*) ***{*** *Bloc d'instructions 1* ***}*** ***else*** ***{*** *Bloc d'instructions 2* ***}***

Les actions qui suivent le *Alors* ou le *Sinon* peuvent être :

- Une simple instruction
- Une suite d'instructions
- Une autre structure alternative
- Une autre structure répétitive

7.1.3. ***Forme à choix :***

Une structure de contrôle conditionnelle est dite à choix lorsque le traitement dépend de la valeur que prendra le sélecteur, Ce scalaire doit être de type scalaire (*entier ou caractère*). Cette structure conditionnelle est appelée aussi à **choix multiple** ou **sélective** car elle sélectionne entre plusieurs choix à la fois, et non entre deux choix alternatifs (le cas de la structure **Si**..**Sinon**).

Vocabulaire et syntaxe:

Algorithme	***Langage C***
SELON (sélecteur) **FAIRE** **Cas <liste de valeurs-1>** : <suite d'action (s)-1> **Cas <liste de valeur-2>** : <suite d'action (s)-2> ……. **SINON** : <suite d'action (s)-n> **FIN SELON**	**switch** (*expression*) ***{*** **case** *constante-1*: *liste d'instructions 1* **break;** **case** *constante-2*: *liste d'instructions 2* **break;** ... **case** *constante-n*: *liste d'instructions n* **break;** **default:** *liste d'instructions 1* **break;** ***}***

Si la valeur de « *expression ou sélecteur»* est égale à l'une des *constantes*, la *liste d'instructions* correspondant est exécutée. Sinon la *liste d'instructions* correspondant à « *Sinon ou default* » est exécutée.

Important : L'instruction *default est facultative.*

Exemples :

Valeur absolus d'un entier	

Algorithme Absolus
Variables X : entier
Début
Lire (x)
Si (x < 0) **alors**
x← -x
fin si
Ecrire ('la valeur absolus est ',x)
Fin
	#include<stdio.h>
	int main()
	{
	int x;
	printf("DONNER LA VALEUR DU X: ");
	scanf("%d",&x);
	if (x<0)
	{
	x= -x;
	}
	printf("la va leur absolus de X est %d \n", x);
	}

Catégorie âge	

Algorithme cat_age
Variables age : entier
Début
Lire (age)
Si (age < 18) **alors**
Ecrire ('vous êtes mineur')
Sinon
Ecrire ('Vous êtes majeur')
Fin si
Fin
	#include<stdio.h>
	int main()
	{
	int age;
	printf("DONNER VOTRE AGE: ");
	scanf("%d",&age);
	if (age<18) **{**
	printf("Vous êtes mineur \n");
	}
	else {
	printf("Vous êtes majeur \n");
	}
	}

Compte de personnes	

Algorithme compte_pers
Variables Nbr_Per : entier
Début
Lire (Nbr_Per)
Selon Nbr_Per **Faire**
Cas 1 : Ecrire ('vous êtes un Monôme')
Cas 2 : Ecrire ('vous êtes un Binôme')
Cas 3 : Ecrire ('vous êtes un Trinôme')
Sinon
Ecrire ('Vous êtes un Groupe')
FinSelon
Fin
	#include<stdio.h>
	int main()
	{
	int Nbr_Per;
	printf("DONNER LE NOMBRE DE PERSONNE: ");
	scanf("%d",&Nbr_Per);
	switch (Nbr_Per)
	{
	case 1: printf ("Vous etes un Monome \n");
	break;
	case 2: printf ("Vous etes un Binome \n");
	break;
	case 3: printf ("Vous etes un Trinome \n");
	break;
	default:
	printf ("Vous etes un Groupe \n");
	break;
	}
	}