Les structures de contrôle

Tests et itérations

Note : on recherchera toujours à se ramener au cas le plus simple.

En particulier, on évitera le cas Alors Rien suivi de Sinon (instructions)

Les tests peuvent être imbriqués

Algorithme	**En C**

Si expression booléenne vraie
Alors
instruction 1
instruction2
Sinon rien
Fin Si
	if (expression booléenne)
	{
	instruction1;
	instruction2;
	}

Algorithme	**En C**

Si expression booléenne
Alors
instruction 1
instruction2
Sinon
instruction3
instruction4
...
FinSi
	if (expression booléenne)
	{
	instruction1;
	instruction2;
	}
	else
	{
	instruction3;
	instruction4;
	...
	}

Algorithme	**En C**

Si expression booléenne
Alors
instruction 1
instruction2
Sinon si expression booléenne vraie
instruction3
instruction4
...
Sinon
instruction3
instruction4
...
FinSi
	if (expression booléenne)
	{
	instruction1;
	instruction2;
	}
	else if (expression booléenne)
	
	{
	instruction3;
	instruction4;
	...
	}
	else
	{
	instruction3;
	...
	}

Le choix multiple

Ce choix est très restrictif :

1 seule expression ou variable ne peut être testée, de type **entier** ou **char,**
sinon des **constantes calculées** (mais on peut le faire dans expression)**.**

Algorithme	**Langage C**

Choisir expression **parmi**
 	
constante1: sequence1
 	
constante2, constante3: sequence2
 	
....
 	
constanteN: sequenceN
 	
sinon sequence N+1
fin
	switch (expression)
	{
	case constante1: sequence1;
	break;
	case constante2,constante3: sequence2;
	break;
	....
	
	case constanteN: sequenceN;
	break;
	default:sequence N+1;
	}

La boucle Pour

Algorithme	En C
Pour variable de **valeur initiale** à **valeur finale** par pas **de** instructions **Fin pour**	**for (**variable= v_initiale; condition de sortie; modifier variable**)** **{** instructions; **}**

Remarques :

Il est interdit, en programmation structurée, de modifier la variable compteur, ou la condition de sortie

Exemples à connaître :

Piége :

```
for (i=0;i<12;i++);
  {
        fonction();
   }
```

Ne MARCHE PAS à cause du ' ; ' après la parenthèse de la ligne contenant le for !!

Compte le nombre de caractères lus sur l'entrée standard au moyen de la fonction getchar( )

```
for ( nb = 0 ; getchar( )  != EOF ;  nb++ )
```

Boucle Répéter - tant-que

Commence par répéter : cette structure sera toujours exécutée au moins une fois.

L'**évaluation de la variable compteur** ne se fera qu'**après chaque tour de boucle**

Algorithme	En C
Répéter instruction1 instruction2 **Tant que (expression booléenne vraie)**	**do** **{** instruction1; instruction2; **}** **while** (expression booléenne)

Boucle Tant que ... répéter

Cette boucle s'utilisera chaque fois qu'on ne pourra pas utiliser correctement la boucle pour et qu'on ne sera pas certain d'utiliser la boucle au moins une fois.

Le corps de la boucle Tant Que pourra ne pas être exécuté (ne sera pas forcément exécuté, au moins une fois)

On n'entrera dans la boucle que si la condition de passage est vraie

Algorithme	En C
Tant Que condition de passage vraie instruction1 instruction2 **Fin tant que**	**while** (expression booléenne) **{** instruction1 instruction2 **}**

Exemple : racines.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

static void usage(char *s);

int
main(int argc, char *argv[])
{
    if (argc != 4)
        usage(argv[0]);
    else
    {
      double a = atof(argv[1]);
      double b = atof(argv[2]);
      double c = atof(argv[3]);
      double delta = b*b - 4*a*c;

      if (delta < 0)
          printf("Pas de racine réelle\n");
      else if (delta == 0)
          printf("Une racine double : %g\n", -b/(2*a));
      else
      {
          double racine_de_delta = sqrt(delta);

          printf("Deux racines : %g et %g\n",
               (-b - racine_de_delta)/(2*a),
               (-b + racine_de_delta)/(2*a));
      }
    }
    return EXIT_SUCCESS;
}

static void
usage(char *s)
{
    fprintf(stderr, "Usage: %s \"a\"  \"b\" \"c\" \n", s);
    exit(EXIT_FAILURE);
}
```

Choix de la boucle la plus adaptée au problème :

Se poser toujours les mêmes questions, dans l'ordre suivant :

Pour ce traitement répétitif :

Nombre d'itérations connu ?

Si la réponse est **Oui --> boucle Pour**

Si la réponse est **Non** : doit on exécuter ce traitement au moins une fois

au moins une itération : Répéter ... Tant Que

on ne sait pas : Tant Que ... Répéter

Contrôle d'acquisition sans message d'erreur

Quelque chose doit être entré au clavier. Et en cas d'erreur, recommencer systématiquement l'acquisition

Répéter
    Ecrire (entrer quelque chose)
    Lire (ce qui est entre)
Tant que (entrée non correcte)

Contrôle d'acquisition avec message d'erreur

On contrôle une information saisie au clavier.
En cas d'erreur on la signale, et on invite à une nouvelle saisie

Ecrire (entrer quelque chose)
Lire (ce qui est entre)
Tant que (condition non satisfaite)
    Ecrire (incorrect, recommencer)
    Ecrire (entrer quelque chose)
    Lire (ce qui est entre)
Fin Tant que

```c
/* caracteres.c */
#include <ctype.h>
#include <stdlib.h>
#include <stdio.h>

int
main(void)
{
    for (;;)
    {
        char c = getchar();

        if (c == '\n')
            break;
        switch (c)
        {
          case 'é' :
            putchar('É');
            break;

          case 'è' :
            putchar('È');
            break;

          case 'ê' :
            putchar('Ê');
            break;

          default:
            putchar(toupper(c));
            break;
        }
    }
    putchar('\n');
    return EXIT_SUCCESS;
}
```

Lien : http://dept-info.labri.u-bordeaux.fr/~achille/MPC-3/2T/