

CHAPITRE N° 2

STRUCTURES ALTERNATIVES

Introduction

- Le programme commence l'exécution à la fonction **main()**.
- Les instructions de la fonction **main()** sont ensuite exécutées de haut en bas, ligne par ligne. Cependant, cet ordre est **rarement rencontré** dans le vrai programme en C.
- L'ordre d'exécution dans la fonction **main()** peut être subdivisé.
- La modification de l'ordre dans lequel les instructions sont exécutées est appelée contrôles de programme.
- Accompli en utilisant des instructions de contrôle de programme, nous pouvons donc contrôler les flux du programme.
- Il existe trois types de contrôles du programme:
 1. Séquences et Blocs
 2. Structures de sélection telles que **if**, **if-else**, imbriqué **if**, **if-if-else**, **if-else-if** et **switch-case-break**.
 3. Répétitions (boucles) comme **for**, **while** et **do-while**.
- Certaines fonctions et mots-clés C peuvent également être utilisés pour contrôler le programme.

Prenons l'exemple suivant:

```
#include <stdio.h> // put stdio.h file here
```

```
int main(void)
```

```
{
```

```
    float paidRate = 5.0, sumPaid, paidHours = 25;
```

```
    sumPaid = paidHours * paidRate;
```

```
    printf("Paid sum = $%.2f \n", sumPaid);
```

```
    return 0;
```

```
}
```

printf("...")
definition

Appel à printf()

Retour à la fonction main() de printf()

- Un point d'entrée et un point de sortie.
- Conceptuellement, une structure de contrôle comme celle-ci signifie une exécution de séquence.

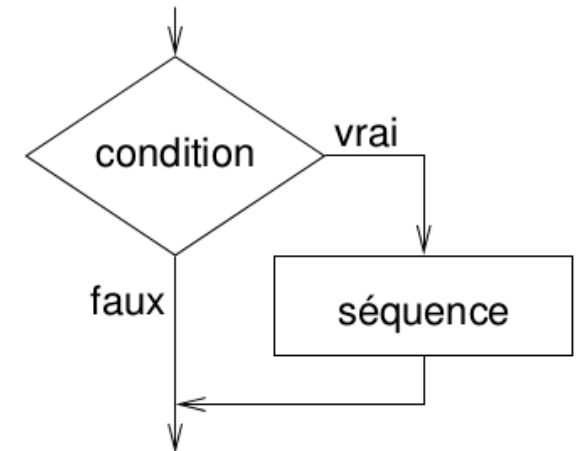
<code>float paidRate = 5.0, sumPaid, paidHours = 25;</code>	S1
<code>sumPaid = paidHours * paidRate;</code>	S2
<code>printf("Paid sum = \$%.2f \n", sumPaid);</code>	S3
<code>return 0;</code>	S4



Structure : if, if-else, if-else-if

- Le programme doit sélectionner parmi les options données pour l'exécution.
- Au moins 2 options, peut être plus de 2.
- Une option sera sélectionnée en fonction du résultat de l'évaluation de la condition: **VRAI** ou **FAUX**.
- En partant de la syntaxe **if** la plus basique,

if (condition)	if (condition)
Instructions;	{Instructions;}
Instruction_suivante;	Instruction_suivante;



1. (**condition**) est évaluée.
2. Si **Vrai** (non-zero) l'instruction s'exécute.
3. Si **false** (zero), l'instruction `Instruction_suivante` s'exécute en sautant le bloc d'instructions de `if`.
4. Ainsi, pendant l'exécution, en fonction de certaines conditions, certains blocs ont été ignorés.

Programme d'exemple

structurelf.c

```
#include <stdio.h>
#include <locale.h>
void main()
{
    setlocale(LC_CTYPE, "");
    int x, y;

    printf("Entrer une valeur pour x:");
    scanf("%d", &x);
    printf("Entrer une valeur pour y:");
    scanf("%d", &y);

    if (x>y)
    {
        printf("x est supérieur à y.\n");
    }
    if (x<y)
    {
        printf("x est inférieur à y.\n");
    }
    if (x==y)
        printf("x est égal à y.\n");
    printf("FIN DU PROGRAMME.");
}
```

Sortie :

```
Entrer une valeur pour x:15
Entrer une valeur pour y:15
x est égal à y.
FIN DU PROGRAMME.
```

Structure : if, if-else, if-else-if

if (condition)	if (condition)
Instruction 1;	{ Bloc d'insctructions 1;}
else	else
Instruction 2;	{ Bloc d'insctructions 2;}
Instructions suivantes;	Instructions suivantes;

Explication:

1. La (**condition**) est évaluée.
2. S'elle est **VRAIE** , Instruction 1 s'est exécute, sinon, s'elle est **FAUSSE**, Instruction 2 s'est exécute.
3. Elles sont mutuellement exclusifs, ce qui signifie que l' Instruction 1 s'est exécutée **ou** l'Instruction 2 , mais pas les deux.
4. Les Instruction 1 et les Instruction 2 peuvent être un bloc de codes et doivent être placées entre accolades **{}**.

Programme d'exemple

ageMajeurMineur.c

```
#include <stdio.h>
#include <locale.h>
void main()
{
    setlocale(LC_CTYPE, "");
    int age;

    printf("Entrer votre âge :");
    scanf("%d", &age);

    if (age >= 18)
        printf("Vous êtes majeur (%d >= 18).\n", age);
    else
        printf("Vous êtes mineur (%d < 18).\n", age);

    printf("FIN DU PROGRAMME.");
}
```

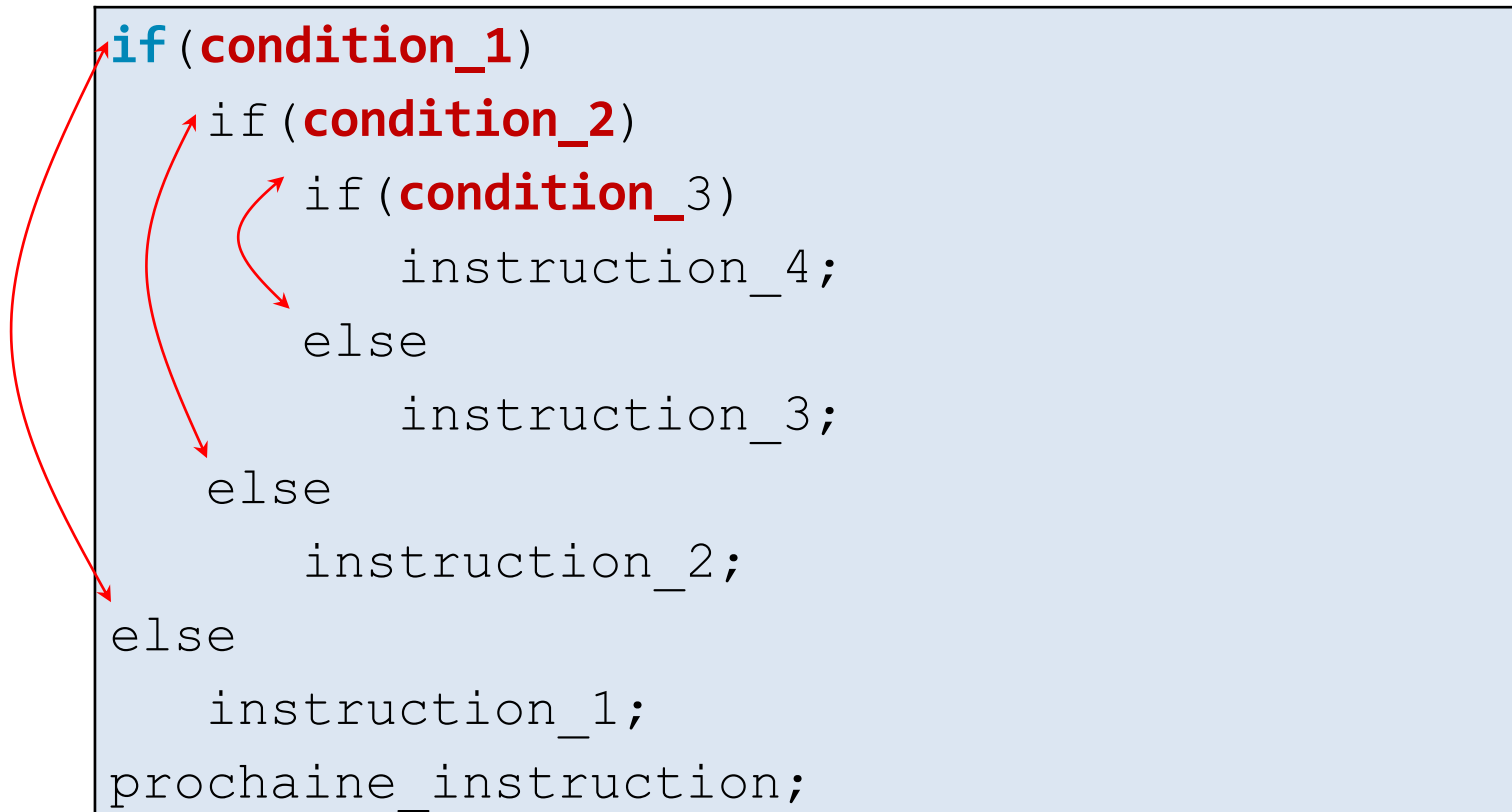
Sortie :

```
Entrer votre âge : 19
Vous êtes majeur (19 >= 18).
FIN DU PROGRAMME.
```

```
Entrer votre âge : 15
Vous êtes mineur (15 < 18).
FIN DU PROGRAMME.
```


Structure : if, if-else, if-else-if

- Les constructions **if-else** peuvent être imbriquées (*placées les unes dans les autres*) à n'importe quelle profondeur.
- Formes générales: **if-if-else** et **if-else-if**.
- Les constructions **if-if-else** ont la forme suivante (*exemple de 3 niveaux de profondeur*),



```
if (condition_1)
    if (condition_2)
        if (condition_3)
            instruction_4;
        else
            instruction_3;
    else
        instruction_2;
else
    instruction_1;
prochaine_instruction;
```

Structure : if, if-else, if-else-if

- Dans cette forme imbriquée, la **condition_1** est évaluée. S'elle est FAUSSE, `instruction_1` est exécutée et l'intégralité de l'instruction if imbriquée est terminée.
- S'elle est VRAIE, le contrôle passe au second **if** (dans le premier **if**) et la **condition_2** est évaluée.
- S'elle est FAUSSE, `instruction_2` est exécutée; sinon, le contrôle passe au troisième **if** (dans le second **if**) et la **condition_3** est évaluée.
- S'elle est FAUSSE, `instruction_3` est exécutée; sinon, `instruction_4` est exécutée. L'`instruction_4` (la plus interne) ne sera exécutée que si toutes les instructions **if** sont **VRAIES**.

Encore une fois, une seule des instructions est exécutée, les autres seront ignorées.

- `instructions_x` peut être un bloc de codes et doit être placé entre accolades.

Programme d'exemple

Equation 2 eme degré.c

```
#include<stdio.h>
#include <locale.h>
main(){
    setlocale(LC_CTYPE,"");
    float a, b, c, delta, x1, x2;
    printf ("-->Saisissez les valeurs a, b et c \n de l'équation ax²+bx+c=0 : \n");
    printf (" -> Donner a : "); scanf ("%f",&a);
    printf (" -> Donner b : "); scanf ("%f",&b);
    printf (" -> Donner c : "); scanf ("%f",&c);
    if (a==0){
        printf(" \n Equation du premier degré \n ");
        if (b!=0 ) printf("La solution est: %f .", -c/b);
        else printf (" Pas de solution");
    }
    else{
        delta = b*b-4*a*c;
        if (delta > 0) {
            x1= (-b -sqrt(delta))/ (2*a);
            x2= (-b+ sqrt(delta))/(2*a) ;
            printf("les solutions sont %f et %f \n",x1,x2 );
        }
        else{
            if (delta == 0) printf ( "La solution est : ", -b/(2*a));
            else printf (" Il n'y a pas de solutions réelles !!\n");
        }
    }
}
```

Sortie :

```
-->Saisissez les valeurs a, b et c
    de l'équation ax²+bx+c=0 :
-> Donner a : 0
-> Donner b : 2
-> Donner c : 5
```

```
Equation du premier degré
La solution est: -2.500000 .
```

```
-->Saisissez les valeurs a, b et c
    de l'équation ax²+bx+c=0 :
-> Donner a : 1
-> Donner b : 8
-> Donner c : 4
les solutions sont -7.464102 et -0.535898
```

Structure : if, if-else, if-else-if

- L'instruction **if-else-if** a la forme suivante (exemple à 3 niveaux).

```
if(condition_1)
    instruction_1;
else if (condition_2)
    instruction_2;
else if (condition_3)
    instruction_3;
else
    instruction_4;
Instruction_suivante;
```

Structure : if, if-else, if-else-if

- **condition_1** est évaluée. Si elle est VRAIE, `instruction_1` est exécutée et toute l'instruction est terminée et l'exécution se poursuit sur l'`instruction_suivante`.
- Si la **condition_1** est FAUSSE, le contrôle passe au prochain **else-if** et la **condition_2** est évaluée.
- Si elle est VRAIE, `instruction_2` est exécutée et tout le système s'est arrêté. Si elle est FAUSSE, la prochaine **else-if** est testée.
- Si la **condition_3** est VRAIE, `instruction_3` est exécutée; sinon, `instruction_4` est exécutée.

Notez qu'une seule des instructions sera exécutée, les autres seront ignorées.

- `instruction_x` peut être un bloc d'instructions et doit être placé entre accolades { }.

Structure : switch-case-break

- Le contrôle de programme de sélection le *plus flexible*.
- Permet au programme d'exécuter différentes instructions basées sur une condition ou une expression pouvant avoir plus de deux valeurs.
- Également appelées instructions à choix multiples.
- L'instruction **if** se limitait à évaluer une expression qui ne pouvait avoir que deux valeurs logiques: **TRUE** ou **FALSE**.
- Si plus de deux valeurs, doivent utiliser **if** imbriquée.
- L'instruction **switch** rend cette imbrication *inutile*. Elle utilise **case** et **break**.

```
switch(condition)
{
    case template_1 :
        statement(s);
        break;
    case template_2 :
        statement(s);
        break;
    case template_3 :
        statement(s);
        break;
    ...
    case template_n :
        statement(s);
        break;
    default : statement(s);
}
next_statement;
```

Structure : **switch-case-break**

- On évalue la (**condition**) et compare sa valeur avec chaque étiquette **case**.
- Si une correspondance est trouvée entre (**condition**) et l'un des étiquettes, l'exécution est transférée à la ou aux instructions qui suivent l'étiquette de **case**.
- Si aucune correspondance n'est trouvée, l'exécution est transférée à la ou aux instructions suivant l'étiquette par **default** (facultative).
- Si aucune correspondance n'est trouvée et qu'il n'y a pas d'étiquette **default**, l'exécution passe à la première instruction suivante en fermant l'accolade de l'instruction **switch**, qui est l'instruction `next_statement`.
- Pour vous assurer que seules les instructions associées au modèle correspondant sont exécutées, incluez un mot-clé **break** si nécessaire, qui met fin à l'ensemble de l'instruction **switch**.
- La ou les instructions peuvent être un bloc de code entre accolades.

Programme d'exemple

choixOperation.c

```
#include<stdio.h>
#include <locale.h>
main()
{
    setlocale(LC_CTYPE,"");
    char operation ;
    short int a=42, b=-7 ;
    printf( "Choisir une opération (+ ou - ou *) : " ) ;
    rewind(stdin) ; /* vide le tampon clavier */
    operation = getchar() ;
    switch (operation)
    {
        case '+': //pour short int
            printf("\n a + b = %hd", a+b) ;
            break ;
        case '-':
            printf("\n a - b = %hd", a-b) ;
            break ;
        case '*':
            printf("\n a * b = %hd", a*b) ;
            break ;
        case '/':
            printf("\n a / b = %hd", a/b) ;
            break ;
        default :
            printf("\n Mauvais choix !") ;
    }
}
```

Sortie :

Choisir une opération (+ ou - ou *) : *

a * b = -294

Choisir une opération (+ ou - ou *) : /



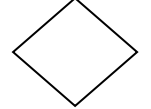


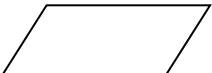
a / b = -6

Les différences entre **if** imbriquées et **switch**

- Les différences entre **if** imbriquées et **switch** :
 1. **switch-case** permet l'exécution de plus d'une alternative alors que l'instruction **if** ne le fait pas. En d'autres termes, les alternatives dans une instruction **if** sont mutuellement exclusives alors qu'elles peuvent ou non l'être dans le cas de **switch-case**.
 2. **switch** peut uniquement effectuer des tests d'égalité impliquant des constantes entières (ou caractères), tandis que l'instruction **if** permet une comparaison plus générale impliquant également d'autres types de données.
- Lorsqu'il y a plus de 3 ou 4 conditions, utilisez l'instruction **switch-case-break** plutôt qu'une longue instruction **if** imbriquée.
- Lorsqu'il y a plusieurs options à choisir.
- Lorsque la condition de test n'utilise que des constantes entières (ou caractères).

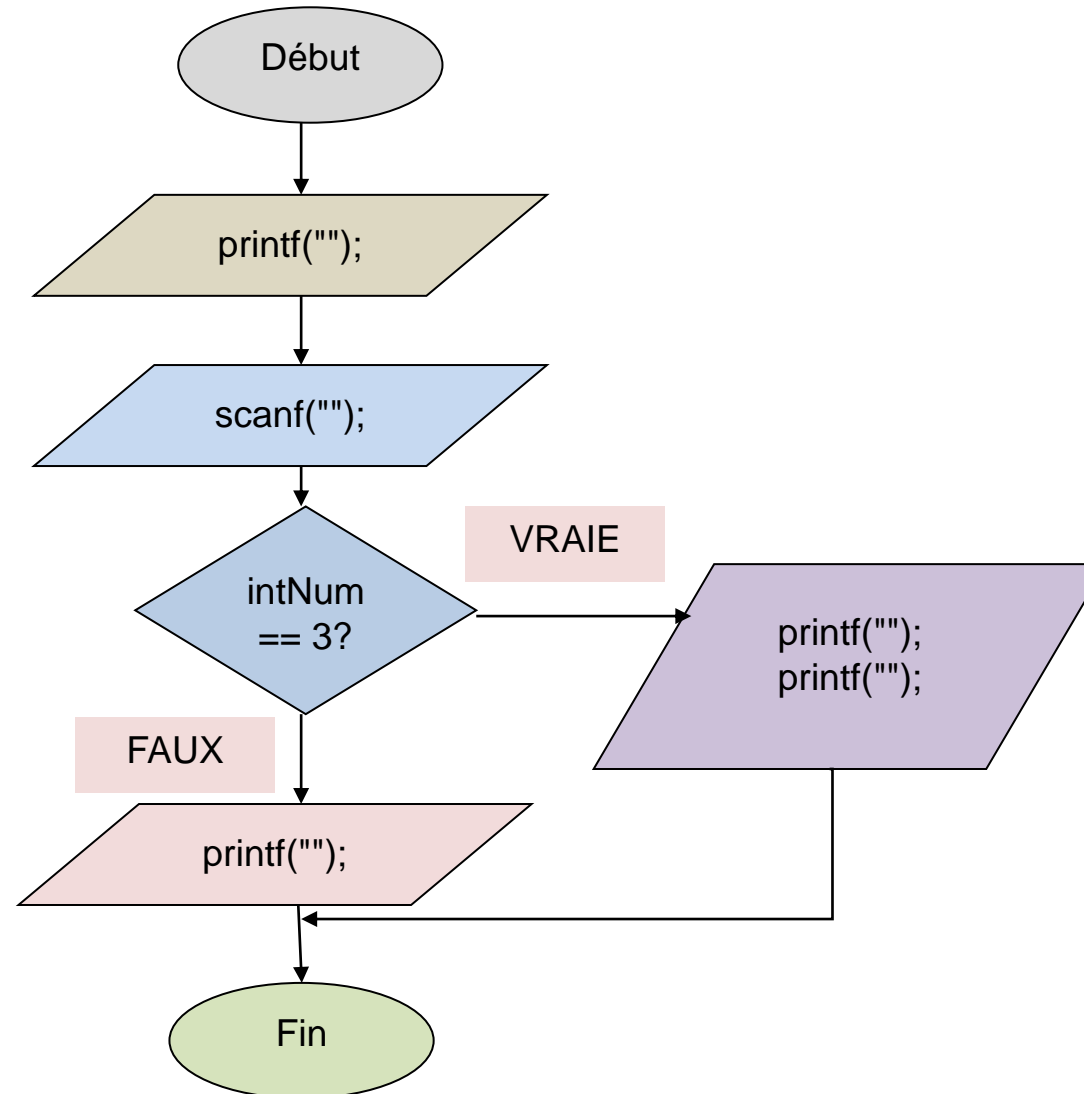
Une histoire d'organigramme

- Une représentation graphique d'un algorithme.
- Dessiné à l'aide de certains symboles tels que des rectangles, des diamants, des ovales et de petits cercles.
- Ces symboles sont reliés par des flèches appelées lignes de flux.
- Les organigrammes montrent clairement l'ordre d'exécution du programme et décrivent indirectement le fonctionnement des structures de contrôle.

Symbole	Nom	Description
	Rectangulaire ou action	Un processus ou une action comme le calcul et l'affectation
	Ovale	Début ou fin. Indique un algorithme ou un flux de programme terminé
	Diamant ou décision	Indique une décision à prendre telle que OUI / NON, VRAI / FAUX, <, <= etc.
	Lignes de flux	Indique l'ordre des actions à exécuter, en connectant d'autres symboles
	Petit cercle ou connecteur	Indique qu'une partie d'un algorithme complet a continué à partir de la partie précédente ou à continuer à la partie suivante
	Entrée ou sortie	L'entrée ou la sortie telle que l'entrée ou la sortie standard

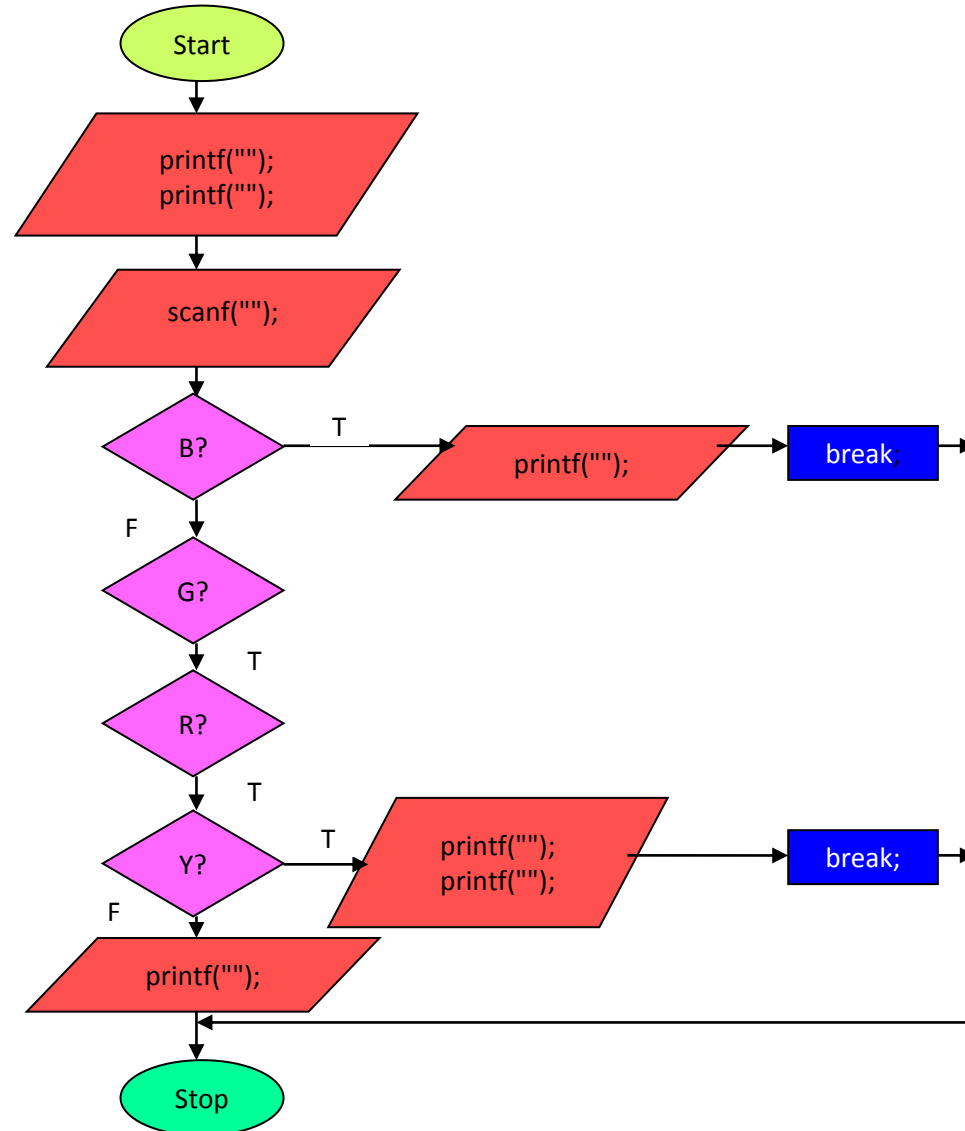
Une histoire d'organigramme

- Les exemples d'organigrammes suivants représentent les instructions de **if** en C.



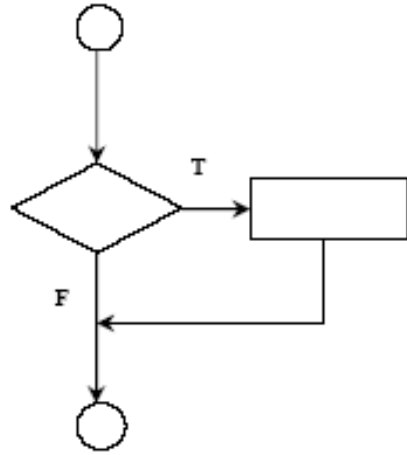
Une histoire d'organigramme

- Les exemples d'organigrammes suivants représentent des constructions de switch-case selection en C.

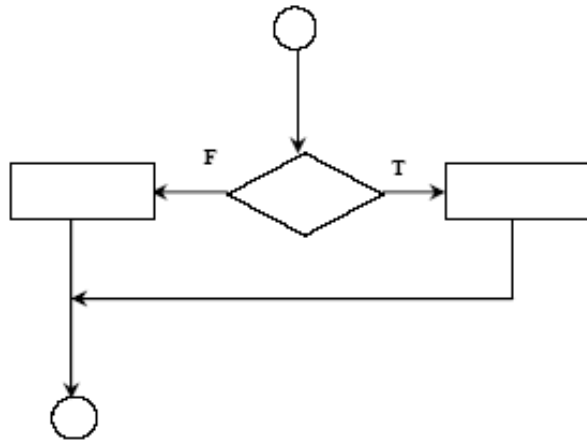


Une histoire d'organigramme

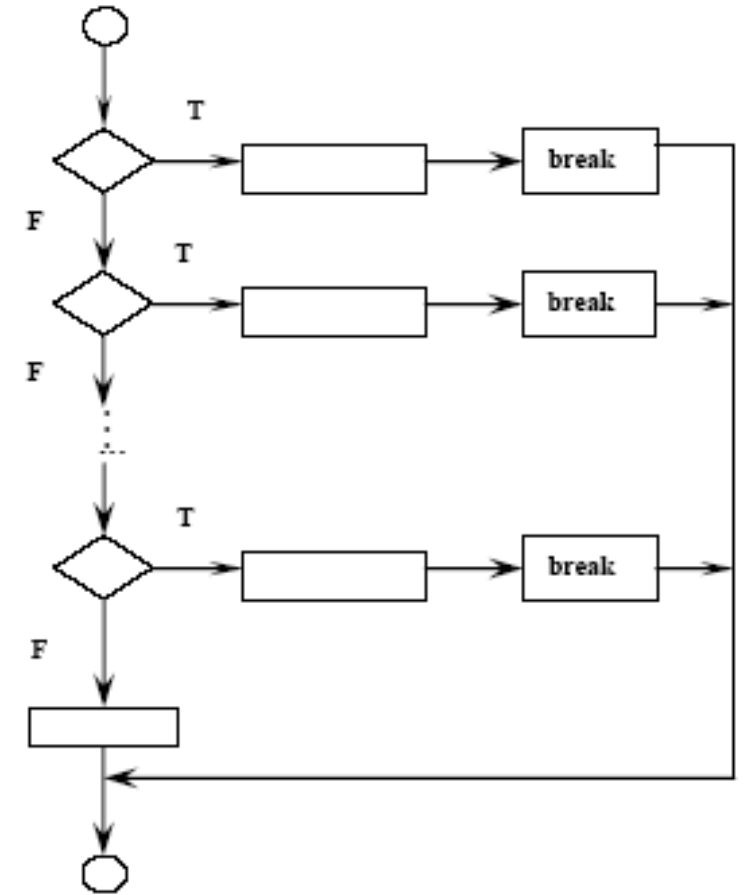
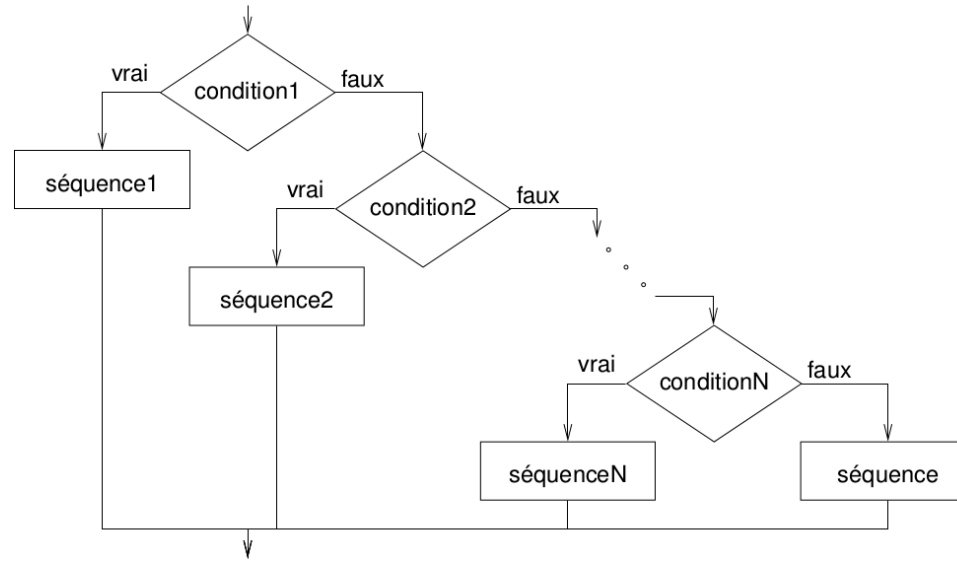
- Les organigrammes **if**, **if-else** et **switch-case-break**



if structure – single selection



if-else structure – double selection



switch structure – multiple selections