Introduction à la programmation (C)

Cours 1 : les bases

« Logistique » de la programmation C

Problème → Prog C → Compilateur Gcc → Exécutable

Fichier texte

toto.c

gcc -o toto toto.c

Langage machine

01110111010001 …

toto

Structure d'un programme

Fichier d'extension .c

Appels à des bibliothèques externes

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

Définition de constantes

```
#define N 10
...
```

Structure d'un programme

Fichier d'extension .c

```
int main ()
{
    …
    return 0 ;

}
```

Types et valeurs

Caractères	Entiers	Réels
char	int	float double
‘a’ ‘b’ ‘\n’ → saut de ligne ‘\t’ → tabulation	1 -5	1.27 -5.1e4 2.34e-5

Types

Valeurs

Variables

Moyen de stocker des données / les récupérer

Variable ⟷ **Emplacement mémoire nommé**

Typée

Syntaxe de base :
type var1, var2 … ;

Au moment de la déclaration :

* obligatoire
* en début de bloc

```
int main ()
{
    int i, j ;
    float x, y ;
    …
}
```

Variables

Affectation (stocker une valeur)

var = val ;

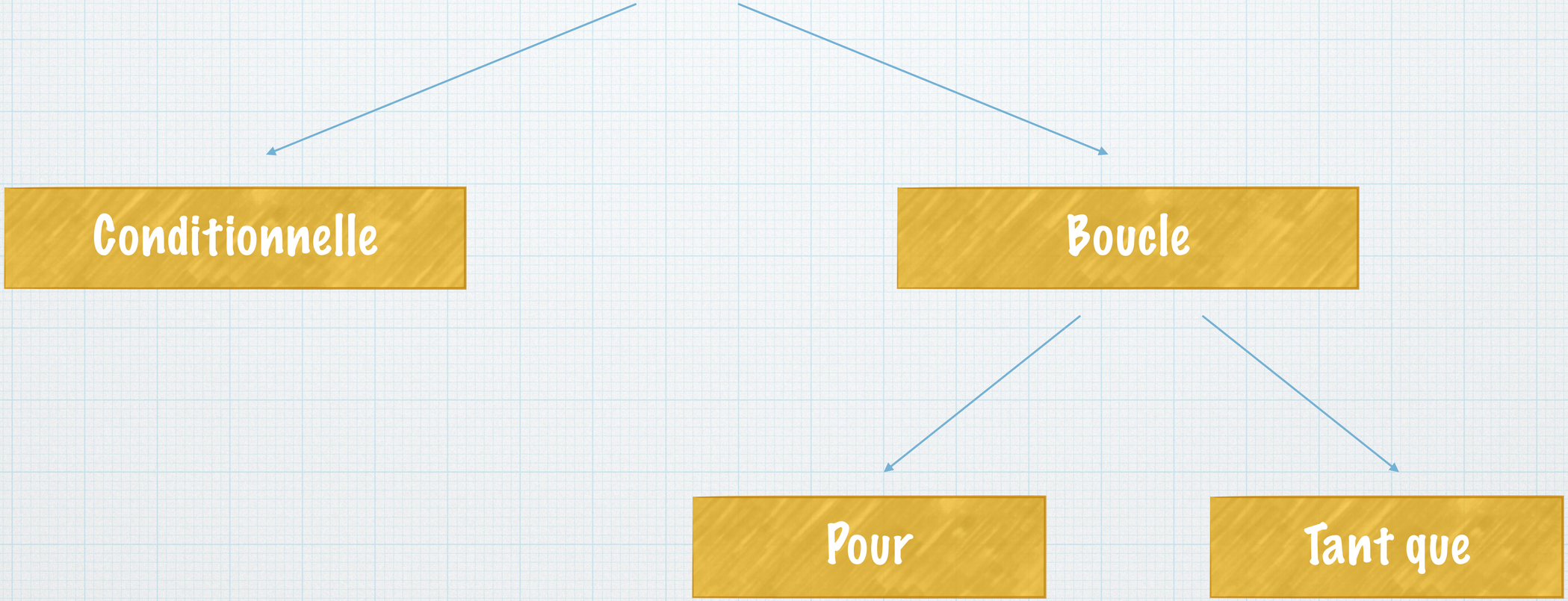
```
i = 2 ;
j = -5 ;
x = 1.25e5 ;
```

Déclaration / initialisation simultanée (début de bloc)

type var = val, … ;

```
int main ()
{
    int i=1, j ;
    float x, y=3.56 ;
    …
}
```

Structures de contrôle



Conditionnelle

```
if (cond)
{
    // code si cond vraie
}
else
{
    // code si cond
fausse
```

Optionnel

```
int main ()
{
    int i=-1, res ;
    if (i > 0)
    {
        res = i ;
    }
    else
    {
        res = -i ;
    }
}
```

Boucle tant que

```
while (cond)
{
    // code à répéter
}
```

```
int main ()
{
    int i=0, res ;
    float x = 1257.5 ;
    while (i*i < x)
    {
        i = i+1 ;
    }
    res = i-1 ;
}
```

Boucle pour

On ne sort jamais d'un for par un break
↓
utiliser while (barème : 0)

Initialisation de la variable de boucle

Condition d'arrêt

```
for (init ; cond ; incr)
{
    // code à répéter
}
```

Instruction d'incrémentation

```
int main ()
{
    int n=10, res=1, i ;
    for (i=1 ; i <= n ; i = i+1)
    {
        res = res * i ;
    }
}
```

Opérateurs logique

ET	&&	(x<2) && (x>-2)
OU	||	(x>=2) || (x<=-2)
EGAL	==	c == ‘a’
NON	!	!((c == ‘a’) && (n<10))
DIFFERENT	!=	c != ‘a’

Entrées/sorties

* Nécessitent d'inclure la bibliothèque
  stdio.h
* Sortie (affichage écran) :
  printf
* Entrée (lecture clavier) :
  scanf

printf

Décrit ce que l'on veut afficher et comment

```
printf("chaine format\n",var1, var2 …) ;
```

Texte simple à afficher

Joker (%) pour indiquer qu'il faut insérer le contenu d'une variable

%d → entier
%f → float
%g → double
%c → char
(%s → chaîne de caractères)

printf

```
int main ()
{
    int n=4, res=1, i ;
    for (i=1 ; i <= n ; i = i+1)
    {
        res = res * i ;
    }
    printf("la factorielle de n = %d vaut %d\n », n, res) ;
}
```

```
> la factorielle de n = 4 vaut 24
>
```

scanf

Lit au clavier et stocke le résultat dans la variable var

```
scanf("chaine format",&var) ;
```

%d → entier

%f → float

%g → double

%c → char

(%s → chaîne de caractères)

& obligatoire
cf. Chapitre 4

scanf

```
int main ()
{
    int n, res=1, i ;
    printf("Entrez la valeur de n : \n") ;
    scanf("%d",&n) ;
    for (i=1 ; i <= n ; i = i+1)
    {
        res = res * i ;
    }
    printf("la factorielle de n = %d vaut %d\n », n, res) ;
}
```

```
> Entrez la valeur de n :
> 5
> la factorielle de n = 5 vaut 120
>
```