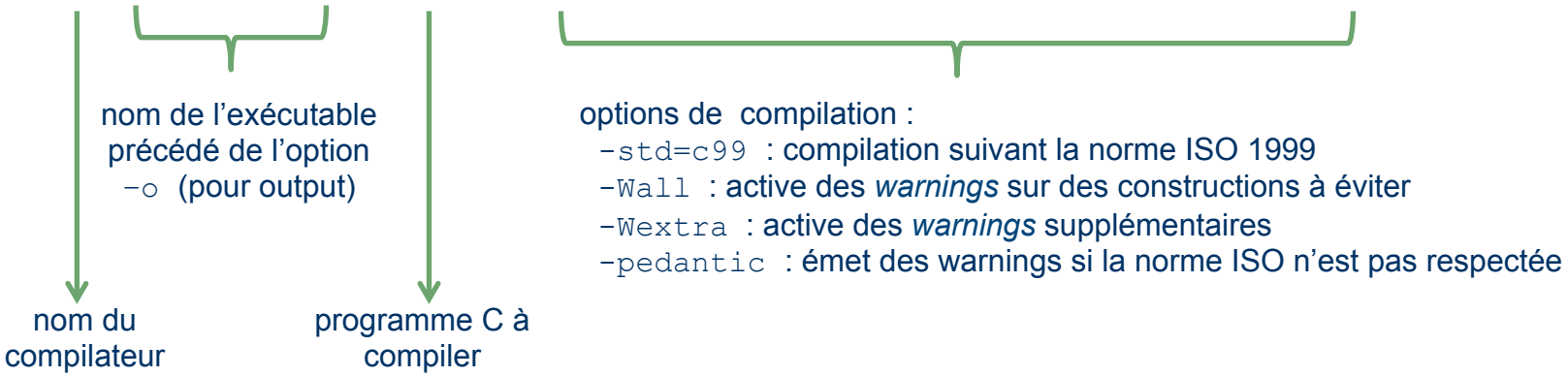
Séance 1

Compilation

- La compilation d'un programme se fait à travers la commande suivante :

```
gcc -o hello hello.c -std=c99 -Wall -Wextra -pedantic
```



Structure d'un programme C

le texte compris entre les symboles `/*` et `*/` est considéré comme un commentaire

pour cette ligne, le texte à droite des symboles `//` est considéré comme un commentaire

```
/*
CONTENU      : Exemple de programme
AUTEUR       : Emmanuel MORIN
CREATION     : 23/07/2013
MODIFICATION : 24/07/2013
*/

// librairies
#include <stdio.h>
#include <stdlib.h>


// corps du programme principal
int main(void)
{
   printf("Bonjour");

   return 0;
}
```

entête du programme

librairies à inclure

corps du programme principal

Utilisation de variables et constantes

- Déclaration et utilisation d'une variable :
  - Exemples :

```
int a;   // a est entier
float b; // b est un réel
char c;  // c est un caractère
```

- Affectation d'une variable :
  - Exemple :

```
int c;
c = 3;
```

- Initialisation d'une variable :
  - Exemple :

```
int c = 3;
```

- Déclaration d'une constante :
  - exemple :

```
const float PI = 3.14;
```

Utilisation de variables

- Portée (ou visibilité) d'une variable
  - **portée globale** : une variable déclarée au début du code (avant le `main` par exemple) sera globale et donc utilisable de n'importe où dans le programme
  - **portée locale** : une variable déclarée à l'intérieur d'un bloc d'instructions (dans le `main` par exemple) aura une portée limitée à ce bloc et sera donc inutilisable ailleurs

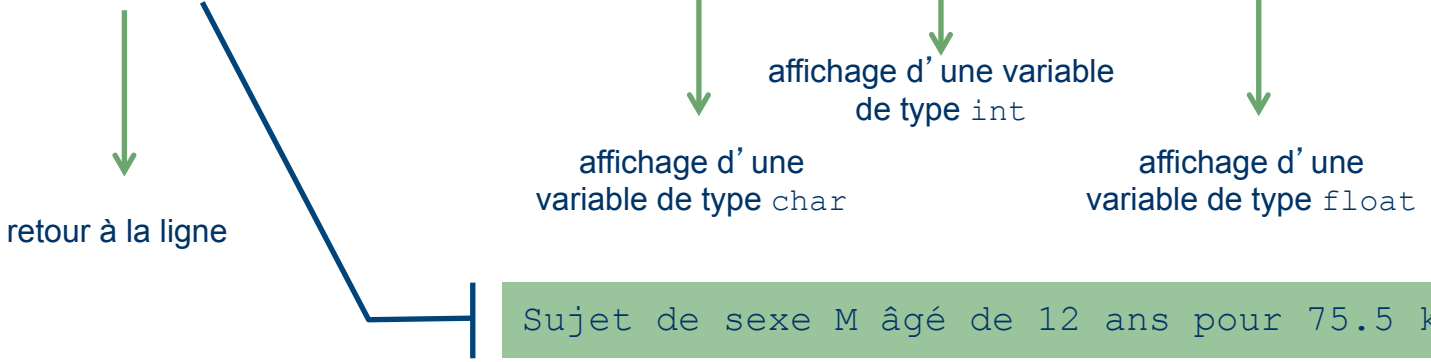
Opérateurs

prudence avec ces opérateurs

- Opérateurs arithmétiques :
  - `+`, `-`, `*`, `/` et `%` (pour le reste de la division entière)
  - `+=`, `-=`, `*=` et `/=` (facilité d'écrire, par exemple `i=i+20;` ⇔ `i+=20;`)
  - `i=i+1;` peut s'écrire `i++;` (post-incrémentation) ou `++i;` (pré-incrémentation)
    - `i=++j;` ⇔ `j=j+1; i=j;`
    - `i=j++;` ⇔ `i=j; j=j+1;`
- Opérateurs logiques :
  - `!` (négation)
  - `&&` et `||` (et vs ou logiques)
- Opérateurs relationnels :
  - `==` et `!=` (tests d'égalité et inégalité)
  - `>`, `>=`, `<` et `<=`

Commande de sortie

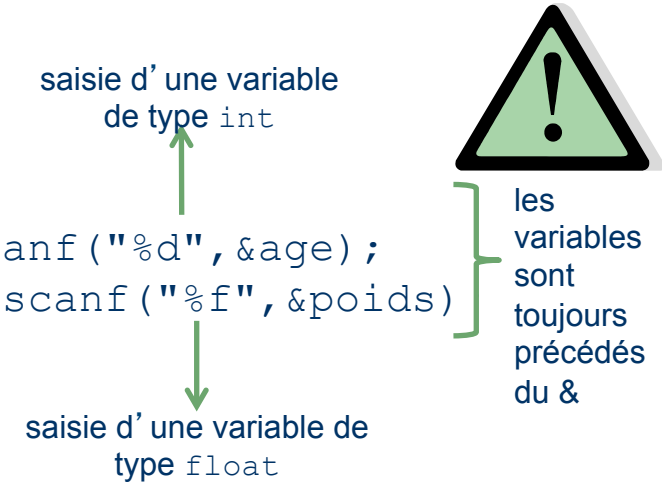
- La sortie sur le terminal (ou dit plus simplement l'affichage) est réalisée par la commande `printf`
  - Exemple avec un seul paramètre :
    ```
    printf("Je suis le texte qui sera affiché !");
    ```
  - Exemple avec plusieurs paramètres :
    ```
    int age = 12;
    float poids = 75.5;
    char sexe = 'M';
    printf("Sujet de sexe %c âgé de %d ans pour %f
    kg\n",sexe,age,poids);
    ```



Commande d'entrée

- L'entrée sur le terminal (ou dit plus simplement la saisie) est réalisée par la commande `scanf`
  - Exemple :

```
int age;
float poids;
printf("Quel est votre âge :"); scanf("%d",&age);
printf("Quel est votre poids :"); scanf("%f",&poids)
```



- Pour la saisie de caractère depuis l'entrée standard, il faut utiliser la commande `getchar`
  - Exemple :

```
char sexe;
printf("Quel est votre sexe : ");
sexe=getchar(); // lit le caractère saisie
getchar();      // lit le retour à la ligne qui suit
```

Structure conditionnelle

- En pseudo langage :

```
si (condition) alors
  série d'instructions 1
fin si
```

```
si (condition) alors
  série d'instructions 1
sinon
  série d'instructions 2
fin si
```

- En langage C :

```
if (condition) {
  série d'instructions 1
}
```

```
if (condition) {
  série d'instructions 1
} else {
  série d'instructions 2
}
```

Si la série d'instructions se limite à une seule instruction il ne sera pas nécessaire de la délimiter avec des accolades

Choix multiples

- En pseudo langage :

```
selon choix
  pour choix1 : série d'instructions 1
  pour choix2 : série d'instructions 2
  pour choix3 : série d'instructions 3
  ...
  autrement : série d'instructions par défaut
fin selon
```

- En langage C :

```
switch choix {
  case choix1 : série d'instructions 1; break;
  case choix2 : série d'instructions 2; break;
  case choix3 : série d'instructions 3; break;
  ...
  default : série d'instructions par défaut; ; break;
}
```

Librairies

- Vous pouvez avoir besoin dans vos programmes de faire appel à des librairies `#include <nom_de_la_librairie>` :
  - `stdbool.h` pour utiliser les booléens
  - `math.h` pour utiliser des fonctions mathématiques (comme `sqrt()` pour la racine carrée)

- Certaines librairies sont statiques ce qui nécessite que code de la librairie doit être inclus dans l'exécutable lors de la compilation :
  - lors de la compilation il faut alors ajouter l'option `-l<bibliothèque>`
  - c'est le cas avec la librairie mathématique qui s'appelle `libm` il faudra alors ajouter l'option de compilation `-lm` (avec `m` comme mathématiques)