

LANGAGE DE PROGRAMMATION: C

F. Karami

Introduction

- **Objectifs du cours**
- **Historique**
- **Intérêt du langage C**
- **Etapas de réalisation du programme**
- **Votre premier programme C**

Introduction : objectifs

- ✓ Concevoir, écrire et exécuter des programmes en langage C
- ✓ Savoir lire, stocker et afficher des données et résultats
- ✓ Savoir utiliser des structures de contrôle (conditions, tests, boucles,...)
- ✓ Savoir définir et utiliser des fonctions
- ✓ Initiation sur des notions non habituels: pointeurs, allocation dynamique, ...

Introduction : historique

- ⦿ Langage de programmation développé en 1970 par **Dennie Ritchie** aux Laboratoires Bell d' AT&T.

Il est l' aboutissement de deux langages :

- ☞ BPCL développé en 1967 par **Martin Richards**.
- ☞ B développé en 1970 chez AT&T par **Ken Thompson**.

Il fut limité à l' usage interne de Bell jusqu' en 1978 date à laquelle Brian Kernighan et Dennie Ritchie publièrent les spécifications définitives du langage :
« **The C programming Language** ».

Introduction : historique

Au milieu des années 1980 la popularité du langage était établie.

De nombreux compilateurs ont été écrits, mais comportant quelques incompatibilités portant atteinte à l'objectif de portabilité.

Il s'est ensuivi un travail de normalisation effectué par le comité de normalisation X3J11 de l'ANSI qui a abouti en 1988 avec la parution par la suite du manuel :

« **The C programming Language – 2^{ème} édition** ».

Introduction : intérêt du langage C

- ☞ Langage polyvalent permettant le développement de systèmes d'exploitation, de programmes applicatifs scientifiques et de gestion.
- ☞ Langage structuré.
- ☞ Langage évolué qui permet néanmoins d'effectuer des opérations de bas niveau (« assembleur d'Unix »).
- ☞ Portabilité (en respectant la norme !) due à l'emploi de bibliothèques dans lesquelles sont reléguées les fonctionnalités liées à la machine.
- ☞ Grande efficacité et puissance.

Introduction : Etapes de la réalisation

1) Création d'un code source en utilisant un éditeur

- Le code source est une série de commandes qui indiquent à l'ordinateur les tâches que vous voulez lui faire exécuter, il est créé à l'aide d'un éditeur.
- La plupart des systèmes d'exploitation contiennent un éditeur. Sous Unix vous pouvez utiliser, ed, ex, edit emacs ou vi. Microsoft Windows vous offre le bloc-notes.

Introduction : Etapes de la réalisation

2) Compilation du code source

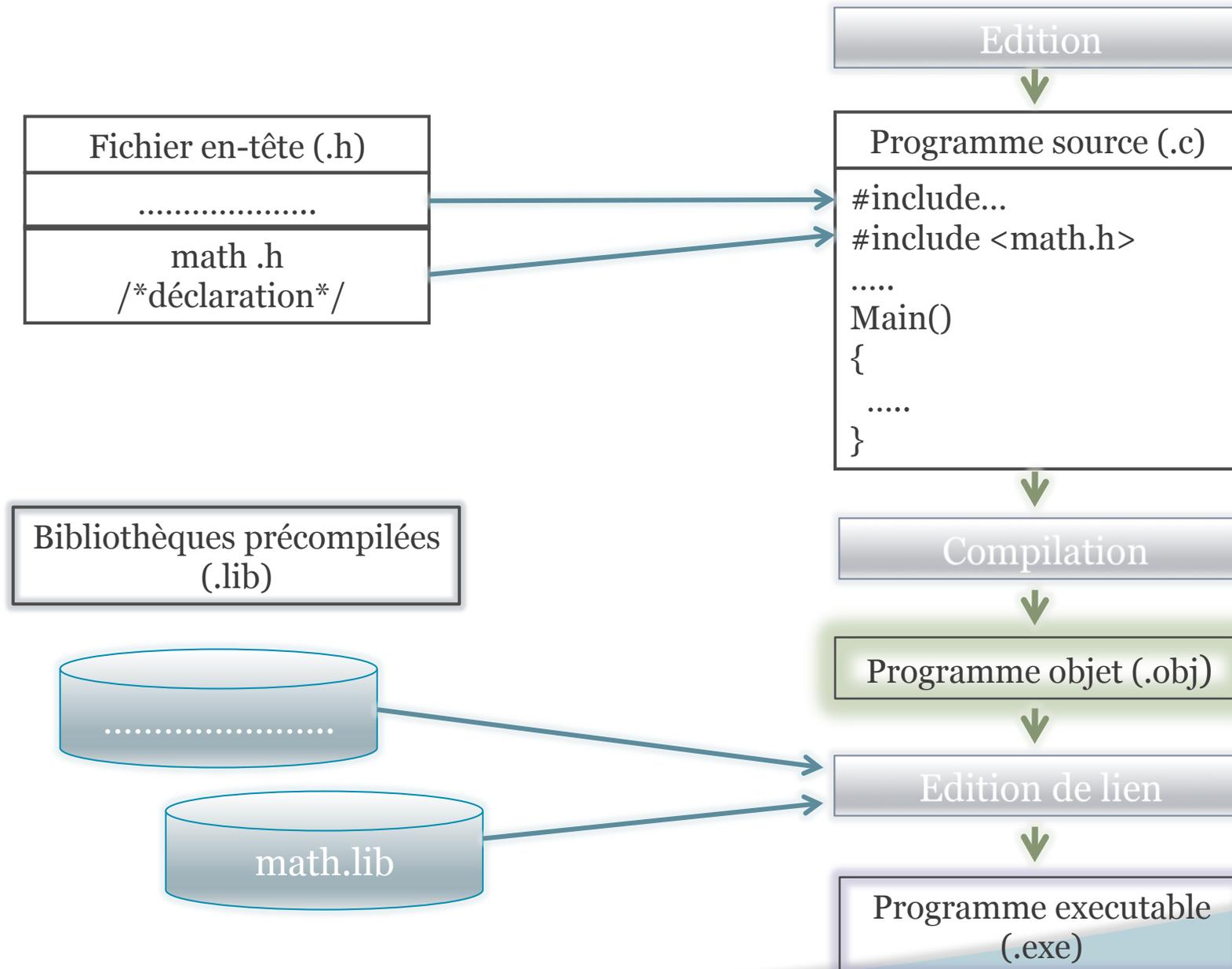
- Votre ordinateur ne peut pas comprendre le code source. Il ne peut comprendre que des instructions binaires dans ce que l'on appelle du **langage machine**.
- Votre programme C doit être transformé en langage machine pour pouvoir être exécuté sur votre ordinateur. Cette opération est réalisée par un **compilateur** qui transforme votre fichier code source en un fichier contenant le code objet (les mêmes instructions en langage machine).
- Chaque compilateur possède sa propre commande pour créer le code objet.
 - Windows: C Microsoft, Turbo C, Turbo C++, Borland C, Borland C++.
 - Unix : utiliser la commande c.
 - Linux et Unix: la commande gcc.

Introduction : Etapes de la réalisation

3) Création du fichier exécutable

- Une partie du langage C est constituée d'une bibliothèque de fonctions contenant du code objet destiné à des fonctions prédéfinies.
- Ces fonctions sont fournies avec votre compilateur. Si votre programme les utilise, le fichier objet obtenu après compilation doit être complété par le code objet issu de la bibliothèque de fonctions.
- Cette dernière étape, appelée **liaison**, fournit le programme exécutable (*exécutable signifie que ce programme peut être exécuté sur votre ordinateur*)

Introduction : Etapes de la réalisation



Introduction : votre premier programme

```
#include<stdio.h>
main()
{
printf( " Bonjour" )
};
}
```

Affichage

Bonjour

Introduction : Etapes de la réalisation

Sous linux: commandes à retenir

- **cc -c essai.c**

Produit le fichier objet `essai.o` qui aura pour nom `a.out`

- **Make essai** ou bien

- **cc -o essai essai.c**

Produit le fichier executable

- **cc -o prog essai.o**

Produit le fichier executable `prog` à partir d'un fichier objet `o`

Eléments de base du langage C

- **Exemple de programme en langage C**
- **Structure d'un programme en Langage C**
- **Les mots-clés**
- **Les types de base**
- **Déclarations des variables simples**

Exemple de programme en langage C

```
#include <stdio.h> //standard input/output
#include <math.h>
#define NFOIS 5
main (){
    int i;
    float x, racx;
    printf("Bonjour\n");
    printf("je vais vous calculer %d racines carrées\n",NFOIS);
    for(i=0;i<NFOIS;i++){
        printf("donnez un nombre:");
        scanf("%f",&x);
        if(x<0.0)
            printf("le nombre %f ne possède pas de racine carrée\n",x);
        else {
            racx=sqrt(x);
            printf("le nombre %f a pour racine carrée :%f\n",x,racx);
        }
    }

    printf("travail terminé Au revoir");
    system(" pause" );
}
```

Bloc

Programme principal

Éléments de base du langage C

Exemple de programme en langage C

```
#include<stdio.h>
main(){
    int i,a;

    printf("Veuillez introduire un nombre: »);
    scanf("%d",&a);
    for(i=a+1;i<=a+10;i++) {
        printf("%d\t",i);
    }

    system(" pause" );
}
```

Structure d'un programme en Langage C

⊙ Les directives à destination du préprocesseur

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

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

```
#define NFOIS 5
```

Programme exécuté automatiquement avant la compilation, il transforme le fichier source à partir d'un certain nombre de directive

- Il s'agit de directive qui seront prises en compte avant la traduction (compilation) du programme. Ces directives doivent être écrites une par ligne, et il est préférable de les placer au début. Les deux premières directives demandent d'introduire (avant compilation) des instructions (en langage C) situées dans les fichiers `stdio.h` et `math.h`. On les appelle fichiers en-têtes.
- La troisième directive définit une constante. Elle demande de remplacer systématiquement, dans toute la suite du programme, le symbole `NFOIS` par `5`

Structure d'un programme en Langage C

- ⊙ La fonction **main()**
 - est un bloc obligatoire d'un programme C. Sa forme la plus simple consiste à saisir son nom, main, suivi de parenthèse () vide et d'une paire d'accolades {}. L'exécution du programme débute à la première instruction de **main()** et se termine avec la dernière instruction de cette fonction.
- ⊙ La définition des variables
 - Une variable est un nom donné à une zone mémoire pour stocker les données en cours d'exécution. En **C** une variable doit être définie avant son utilisation. Sa définition indique son nom au compilateur et le type de données que l'on pourra y stocker.
 - Sa déclaration est de la forme
type *nom_variable* [= <valeur>];
 - Elle peut être déclarée à l'extérieur ou à l'intérieur de la fonction main

Structure d'un programme en Langage C

◎ Notion d'identificateur

- Un identificateur, comme son nom l'indique, permet de donner un nom à une entité du programme (qu'il s'agisse d'une variable ou d'une fonction). Ils sont sujets aux règles suivantes :
 1. Ils sont formés d'une suite de lettres ('a' à 'z' et 'A' à 'Z'), de chiffres (0 à 9) et du signe '_'. En particulier, les lettres accentuées sont interdites ;
 2. le premier caractère de cette suite ne peut pas être un chiffre ;
 3. les identificateurs sont case-sensitive.
- Ainsi, les noms **var1**, **S_i**, et **InitDB** sont des identificateurs valides, tandis que **i:j** ou **1i** ne le sont pas.

Structure d'un programme en Langage C

- ◎ Pour écrire des informations: la fonction **printf**
 - L'instruction **printf("Bonjour\n");** appelle une fonction prédéfinie(printf) qui reçoit un argument "Bonjour\n"

délimiter par des guillemets pour dire que c'est une chaîne de caractères. La notation \n est conventionnelle : elle représente un caractère de fin de ligne, lorsqu'il est envoyé à l'écran, elle provoque le passage à la ligne suivante.

- L'instruction **printf("je vais vous calculer %d racines carrées\n",NFOIS);**

Ressemble à la précédente avec la différence qu'il reçoit un argument que l'on nomme un *format*, c'est comme un guide qui précise comment afficher les informations, dans notre cas le code format **%d** précise qu'il s'agit d'un entier.

Structure d'un programme en Langage C

- ◎ Pour faire une répétition: l'instruction **for**
 - En langage C, il existe plusieurs façon de réaliser une répétition, comme par exemple : **for(i=0;i<NFOIS;i++)**, son rôle est de répéter le bloc délimiter par les accolades, en respectant les consignes suivantes :
 - Avant de commencer cette répétition réaliser : **i=0**
 - Avant chaque nouvelle exécution du bloc, examiner la condition: **i<NFOIS**, si elle est satisfaite, exécuter le bloc indiqué à sa fin réaliser **i++** qui n'est que **i=i+1**
- ◎ Pour lire les informations: la fonction **scanf**
 - **scanf("%f",&x)**: cette instruction n'est qu'appel de la fonction prédéfinie **scanf** dont le rôle est de lire une information au clavier
 - **%f**: le format de la variable
 - **&x**: son adresse

Structure d'un programme en Langage C

⊙ Pour faire le choix: l'instruction *if*

- Les lignes

```
if(x<0.0)
1 printf("le nombre %f ne possède pas de racine carrée\n",x);
else {
2   racx=sqrt(x);
   printf("le nombre %f a pour racine carrée :%f\n",x,racx);
}
```

Constituent une instruction de choix basée sur la condition $x < 0.0$, si cette condition est vraie, on exécute l'instruction 1, si elle est fausse on exécute l'instruction 2.

Structure d'un programme en Langage C

⊙ Les commentaires

- */* un commentaire d'une ligne*/*
- *int a, b, c; /* commentaire sur une partie d'une ligne*/*
- */* un commentaire
Qui s'étend sur plusieurs ligne*/*
- *//cette ligne est en commentaire*

⊙ Conseils:

- Ajouter de nombreux commentaires dans le code source de votre programme, surtout s'il contient des instructions ou fonctions qui pourraient être difficile à comprendre. Vous gagner un temps précieux quand vous aurez à le modifier.

Les mots-clés

- Un certains nombres de mots sont réservés pour le langage C, a évité comme identificateurs, la liste exhaustive est la suivante:

| | | | |
|----------|--------|----------|----------|
| auto | double | int | struct |
| break | else | long | switch |
| case | enum | register | typedef |
| char | extern | return | union |
| const | float | short | unsigned |
| continue | for | signed | void |
| default | goto | sizeof | volatile |
| do | if | static | while |

Les types de base : Les caractères

- ⊙ On utilise le mot-clé char pour désigner une variable de type char. Il s'agit en fait d'un entier codé sur 8 bits interprété comme un caractère utilisé sur la machine (il s'agit en général du code ASCII de ce caractère).
- ⊙ Exemple :
 - `char c1 = 'a' ;` // Déclaration d'une variable c1 de type *char*
// a laquelle on affecte la valeur 'a'
// A noter l'utilisation du simple quote

Les types de base : Les caractères

- Le tableau suivant donne la liste des principaux codes ASCII en décimal

| code | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT |
| 10 | LF | VT | NP | CR | SO | SI | DLE | DC1 | DC2 | DC3 |
| 20 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS |
| 30 | RS | US | SP | ! | " | # | \$ | % | & | ' |
| 40 | (|) | * | + | , | - | . | / | 0 | 1 |
| 50 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; |
| 60 | < | = | > | ? | @ | A | B | C | D | E |
| 70 | F | G | H | I | J | K | L | M | N | O |
| 80 | P | Q | R | S | T | U | V | W | X | Y |
| 90 | Z | [| \ |] | ^ | _ | ' | a | b | c |
| 100 | d | e | f | g | h | i | j | k | l | m |
| 110 | n | o | p | q | r | s | t | u | v | w |
| 120 | x | y | z | { | | } | - | DEL | | |

Les types de base : Les caractères

⊙ Caractères particuliers

- Il existe un certain nombre de caractères particuliers dont les principaux sont résumés dans le tableau suivant :

| | | | | | | |
|------------|--------------------------|-------------------|------------------|----------|----------|----------|
| Caractère | '\n' | '\t' | '\b' | '\'' | '\"' | '\?' |
| Sémantique | <i>retour à la ligne</i> | <i>tabulation</i> | <i>backspace</i> | <i>'</i> | <i>"</i> | <i>?</i> |

⊙ Chaînes de caractères

- Les chaînes de caractères sont vues comme un pointeur sur des caractères et sont donc de type char *.
- Exemple:
 - **char** * chaine = "Hello World !"; // *une chaine de caractère*
 // *noter l'utilisation du double*
 // *quote*

Les types de base : Entiers

- ◎ Le type `int` peut être précisé par des attributs :
 - De longueur :
 - `short` pour des entiers courts(généralement de 8 ou 16 bits)
 - `long` pour des entiers longs(généralement de 32 ou 64bits)
 - La longueur par défaut est généralement de 16 ou 32bits
 - De domaine
 - `signed` pour les entiers relatifs
 - `unsigned` pour les entiers naturels positifs ou nuls

Les types de base : Entiers

- ⊙ On utilise le mot-clé int.
- ⊙ Exemple :
 - */* déclaration la plus courante d'une variable de type int */*
 - **int** a = 14; // la variable a est initialisée à la valeur 14
 - */* Utilisation des précisions (cas le plus général)*/*
 - **short int** b; // b est codé sur 16 bits
 - **int** c; // c est codé sur 16 ou 32 bits
 - **long int** d; // d est codé sur 32 bits
 - *// la possibilité de l'écriture suivante dépend du compilateur*
 - **long long int** e; // e est codé sur 64 bits.
 - */* Précision du signe */*
 - **unsigned long int** n; //n est un entier non signé sur 32 bits

Les types de base : Entiers

- Le tableau suivant regroupe les types entiers standards avec quelques informations supplémentaires :

| Type | Taille mémoire | Intervalle de valeurs |
|-----------------------|----------------|--|
| char | 1 octet | $[-128 ; 127]$ ou $[0, 255]$ |
| unsigned char | 1 octet | $[0 ; 255]$ |
| signed char | 1 octet | $[-128 ; 127]$ |
| int | 2 ou 4 octets | $[-2^{15} ; 2^{15} - 1]$ ou $[-2^{31} ; 2^{31} - 1]$ |
| unsigned int | 2 ou 4 octets | $[0 ; 2^{16} - 1]$ ou $[0 ; 2^{32} - 1]$ |
| short int | 2 octets | $[-2^{15} ; 2^{15} - 1]$ |
| unsigned short int | 2 octets | $[0 ; 2^{16} - 1]$ |
| long | 4 octets | $[-2^{31} ; 2^{31} - 1]$ |
| unsigned long | 4 octets | $[0 ; 2^{32} - 1]$ |
| long long(*) | 8 octets | $[-2^{63} ; 2^{63} - 1]$ |
| unsigned long long(*) | 8 octets | $[0 ; 2^{64} - 1]$ |

Les types de base : Entiers

- ⊙ **Constante entière:** donnée inchangée qui ne peut varier pendant l'exécution d'un programme
 - **Déclaration :** **#define Max 100**
- ⊙ Elle se présente sous forme décimale, octale ou hexadécimal
 - décimale (écriture en base 10) : c'est l'écriture usuelle. Ex : 372 ;
 - octale (base 8) : on commence par un 0 suivi de chiffres octaux. Ex : 0477 ;
 - hexadécimale (base 16) : on commence par 0x (ou 0X) suivis de chiffres hexadécimaux (0-9, a-f). Ex : 0x5a2b, 0X5a2b, 0x5A2b.

Les types de base : Les flottants

- On distingue trois types de flottants : float, double et long double.
 - Exemple : **double** Pi = 3,14159;
- Le tableau suivant donne des informations pour chaque type flottant.

| Type | Taille mémoire | Intervalle de valeurs | Précision |
|-------------|----------------|---------------------------------------|----------------------|
| float | 4 octets | $[1,2 * 10^{-38}; 3,4 * 10^{38}]$ | 6 chiffres décimaux |
| double | 8 octets | $[2,3 * 10^{-308}; 1,7 * 10^{308}]$ | 15 chiffres décimaux |
| long double | 10 octets | $[3,4 * 10^{-4932}; 1,1 * 10^{4932}]$ | 19 chiffres décimaux |

Les types de base : Les flottants

- ⊙ Une constante réelle représente un nombre à virgule flottante (de type `float` ou `double`) sous forme décimale ou exponentielle.
- ⊙ Exemple :

| notation C | correspondance | notation C | correspondance |
|------------|-----------------|------------|----------------|
| 2. | 2 | .3 | 0.3 |
| 1.4 | 1.4 | 2e4 | $2 * 10^4$ |
| 2.e4 | $2 * 10^4$ | .3e4 | $0.3 * 10^4$ |
| 1.4e-4 | $1.4 * 10^{-4}$ | | |

Les types de base : Le type **void**

- ⊙ Toute variable C est typée, de même que toute valeur de retour d'une fonction.
- ⊙ Mais il peut arriver qu'aucune valeur ne soit disponible pour exprimer l'idée de "aucune valeur", pour cela on utilise le mot-clé void.
- ⊙ Ce type est utilisé pour la déclaration de fonctions qui n'ont pas de valeur de retour.

Déclarations des variables simples

- ⊙ Les variables et les constantes sont les données principales manipulées par un programme.
- ⊙ Les déclarations introduisent les variables, fixent leur type et parfois aussi leur valeur de départ(initialisation);

Syntaxe de déclaration:

- `int x,y;`
- `short compteur;`
- `float prix,salaire;`
- `double m;`
- `char s;`

Déclarations des variables simples

◎ Initialisation des variables

- En C, il est possible d'initialiser les variables à la déclaration
- **Exemples:**
 - `int max=123;`
 - `char tab= 'c' ;`
- En utilisant l'attribut **const**, la valeur d'une variable ne change pas au cours du programme: `c` est une constante.
- **Exemple:**
 - `const int max=765;`
 - `const char ch1= 'a' ;`

Le mot-clé typedef

- ⊙ Le mot-clé typedef permet de créer un synonyme pour un type de donnée existant. Par exemple l'instruction :
 - `typedef int entier;`

Crée le synonyme entier pour int.

Vous pouvez ainsi utiliser `entier` pour définir des variables de type `int`, comme dans l'exemple suivant:

- `entier compte; //equivalent int compte;`

Exercices

- ⊙ Quel type de variable convient le mieux pour stocker les valeurs suivantes?
 - L'âge d'une personne
 - Le poids
 - Le rayon d'un cercle
 - Salaire annuel
 - Le prix d'un article
 - La note la plus haute d'un test
 - La température
 - Le gain d'une personne
 - La distance d'une étoile en kilomètre

Sol

⊙ Quel type de variable convient le mieux pour stocker les valeurs suivantes:

- L'âge d'une personne
- Le poids
- Le rayon d'un cercle
- Salaire annuel
- Le prix d'un article
- La note la plus haute d'un test
- La température
- Le gain d'une personne
- La distance d'une étoile en kilomètre

- `unsigned int` age;
- `unsigned int` poids;
- `float` rayon=3;
- `long` salaire_annuel;
- `float` cout=29,95;
- `const int` note_max=100;
- `#define` note_max 100;
- `float` temperature;
- `long` gain=30000;
- `double` distance;

Exercices

- ⦿ Quels sont les noms de variables correctes :
 - a) 123variable
 - b) X
 - c) Score_totale
 - d) Poids_en_#s
 - e) One_0
 - f) Grand-cout
 - g) RAYON
 - h) rayon
 - i) Cela_est_une_variable_pour_stocker_la_largeur

Exercices

⊙ Quels sont les noms de variables correctes :

a) 123variable

➔ b) X

➔ c) Score_totale

d) Poids_en_#s

➔ e) One_0

f) Grand-cout

➔ g) RAYON

➔ h) rayon

➔ i) Cela_est_une_variable_pour_stocker_la_largeur