Qu'est ce que le langage C ?

- Langage de programmation - D. Ritchie (AT&T Bell, 1972)
  - Langage de haut niveau
    - Facile à lire et à comprendre
    - Maintenance des programmes aisée
    - Portabilité des applications sur des plates-formes différentes
  - Langage capable de descendre au niveau matériel
    - Permet de contrôler les différents éléments de l'ordinateur ainsi que les périphériques
      - Langage le plus bas dans la hiérarchie des langages de haut niveau
  - Langage
    - Rapide à apprendre
      - Nombre de commandes et de mots clés réduits
      - Notions d'anglais facilitent son apprentissage
    - Compilé
      - Doit être converti avant exécution en langage machine par un compilateur
    - Normalisé
      - ANSI (American National Standard Institute), 1989
      - ISO (International Standardization Organization), 1990

Premier programme C

- Exemple

```
/* exemple1.c : premier programme */
#include<stdio.h>

int main()
{
   printf("Bonjour : voici un premier programme ! \n");
   return 0;
}
```

  - Remarques
    - C fait la différence entre les lettres minuscules et les lettres majuscules
    - Par convention
      - Les mots clés du langage sont écrits en minuscules
      - Les structures de blocs apparaissent nettement (indentation)

Premier programme C

- Commentaires
  - Permettent de documenter les différentes parties des programmes
    - Facilitent leur compréhension
  - Deux syntaxes
    - Syntaxe 1
      - Commencent par /* et se terminent par */

```
/* ceci est un commentaire */
```

      - Le compilateur ne tient pas compte des caractères figurant entre ces deux marques
      - Les commentaires ainsi marqués peuvent figurer sur plusieurs lignes
    - Syntaxe 2 (non conforme à la norme ANSI)
      - Commencent par //

```
// ceci est un commentaire
```

      - Se terminent par un retour chariot
  - La norme ANSI ne prévoit pas l'imbrication de commentaires

Premier programme C

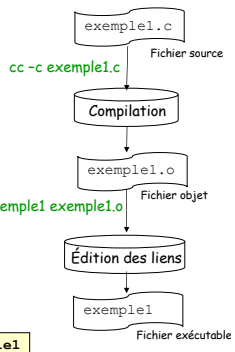
- Directives
  - Commencent par le caractère `#`
  - S'adressent au préprocesseur
    - Programme qui prépare le code C à la compilation
  - `#include <stdio.h>`
    - `#include`
      - Toujours suivie d'un nom de fichier appelé fichier en-tête
      - Ordonne au préprocesseur de
        - Rechercher le fichier indiqué (ici `stdio.h`)
        - L'intégrer à l'emplacement où elle figure dans le fichier source
    - `stdio.h`
      - Fichier en-tête des entrées-sorties standards
        - L'extension `.h` correspond à header (en-tête)
        - Contient de nombreux prototypes et macros de gestions des entrées-sorties des programmes C
    - Utilisation de guillemets doubles ("")
      - Pour rechercher le fichier en-tête dans le répertoire actif
    - Utilisation de chevrons (‹›)
      - Pour rechercher le fichier hors du répertoire actif

Premier programme C

- main()
  - Fonction principale
    - Programme principal dans d'autres langages
    - Point d'entrée du programme
      - Endroit où se trouve la première instruction à exécuter
  - Retourne un entier
    - Grâce à l'instruction `return`
    - Égal à 0 lorsque le programme s'est terminé normalement
      - Une valeur différente de 0 renvoyée par l'instruction `return` indique au système d'exploitation qu'une erreur s'est produite
  - Tout programme C s'arrête lorsque toutes les instructions de la fonction principale ont été traitées
  - Dans l'exemple "exemple1.c"
    - Appel à la fonction C `printf (…)` qui permet d'afficher la chaîne de caractères passée en argument
      - Le caractère `(\n)` correspond à un saut de ligne. Il ordonne à `printf` de passer à la ligne suivante une fois le message affiché

Premier programme C

- Compilation et liaison
  - Fichier source
    - Programme composé d'instructions entrées par le développeur
    - Extension : .c
  - Fichier objet
    - Programme non exécutable car il reste à lui adjoindre des instructions spécifiques
      - Liaison réalisée à l'aide d'un éditeur de liens
    - Extension : .o
  - Fichier exécutable
    - Programme exécutable
    - Extension : .exe




```
> gcc exemple1.c -o exemple1
> ./exemple1
```

Éléments de base du langage

- ✓ Variable
  - » Chaque variable est définie par un nom, une valeur, un type
  - » Avant d'être utilisée, chaque variable doit être déclarée en lui assignant
    - o Un nom (identificateur)
    - o Un type
  - » Identificateur
    - o Suite de caractères alphanumériques, le premier étant alphabétique
    - o Différenciation des lettres majuscules et des lettres minuscules
    - o Pas d'accent, pas d'espace
    - o Mots clés interdits
    - o Exemple
      - – 2essai est un identificateur invalide
      - – nom et NOM sont deux identificateurs différents

Éléments de base du langage

- » Chaque variable est associée à un type définissant
  - o Un ensemble de valeurs possibles pour la variable
  - o Un ensemble d' opérateurs applicables sur ces valeurs
  - o 3 types principaux
    - – Numérique (entier, réel)
    - – Alphanumérique pour caractère ou chaîne de caractères
    - – Booléen pour les valeurs logiques VRAI (true) et FAUX (false)
- » Une variable possède une et une seule valeur à un instant donné qui peut évoluer dans le temps

Éléments de base du langage

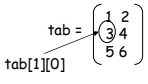
- ✓ Types simples
  - » Types "caractère"
    - o Peuvent être signés ou non (unsigned char)
      - – char : 1 octet ([0; 255] ou [-128; 127])
  - » Types "entier"
    - o Les entiers non signés sont représentés en binaire pur
    - o Les entiers signés sont généralement représentés en complément à 2
      - – short, int, long
  - » Types "flottant"
    - o Se distinguent par le domaine et la précision
      - – Flottant : float
      - – Flottant double précision : double

Éléments de base du langage

Type de base et mot clé	Signification	Taille (en octet)
char	Caractère	1
unsigned char	Caractère non signé	1
short int	Entier court	2
unsigned short int	Entier court non signé	2
int	Entier	4 (sur proc 32 bits)
unsigned int	Entier non signé	4 (sur proc 32 bits)
long int	Entier long	4 ou 8 selon proc
unsigned long int	Entier long non signé	4
float	Flottant (réel)	4
double	Flottant double	8
long double	Flottant double long	10

Éléments de base du langage

- ✓ Déclarations
  - » Toutes les variables utilisées dans un programme C doivent avoir été déclarées avant leur utilisation
  - » Déclaration : association d'une liste d'identificateurs à un type
    - o Déclaration de variables simples
      - – <type><ident>[=<vinit>] {,<ident>[=<vinit>]}*
      - – Exemples : `int minutes=60; double pi=3.1415926535;`
    - o Déclaration de tableaux
      - – Tableau à 1 dimension
        - > <type><ident>[<taille_const>] {, <ident> [<taille_const>]}*
        - > Exemple : `int tab[10];`
      - – Tableau à plusieurs dimensions
        - > <type><ident>[<taille_const>] [<taille_const>] {[<taille_const>]}*
        - > Exemple : `int tab[3][2];`



Éléments de base du langage

- ✓ Les instructions
  - » Le symbole ; (point-virgule) est un terminateur d'instruction
  - » L'affectation
    - o <variable> = <expression>;
      - – `int x;` `x = 1;`
      - – `int tab[10];` `tab[0] = 2;`
    - o Une conversion de type peut intervenir avant cette affectation lorsque les types ne sont pas compatibles
      - – `int c;` `c = 'A';`
    - o L'affectation est aussi une expression de valeur l'expression de droite
      - – `int a, b;` `a = b = 2;`
      - – `int a, b, c, d;` `a = ( b = c + 1) + d;`
    - o Affectation complexe
      - – Pour tout opérateur op ∈ { +, -, *, /, %, <<, >>, &, |, ^}
      - – Exemples
        - > `x += 2;` `// équivaut à x = x + 2;`
        - > `x *= y+1;` `// équivaut à x = x * ( y + 1 );`

Éléments de base du langage

- Exemple

Type	Nom	Valeur
entier	somme	10
réel	moyenne	13,5
booléen	estTerminé	faux
caractère	lettre	'a'
chaîne de caractères	nom	"dupond"

- Traduction
  - `int somme = 10;`
  - `float moyenne = 13.5;`
  - `bool estTermine = false;`
  - `char lettre = 'a';`
  - `char * nom = "dupond";`

Éléments de base du langage

- ✓ Conversion de types (trans-typage)
  - Adaptation d'une valeur pour l'affecter à une donnée d'un autre type
- ✓ Conversion implicite, automatique (à la compilation)
  - A EVITER ➔ respecter le typage !!!

```
int x;
float f;

x = 3.0/2.0;  /*x contient la valeur 1 après l'affectation */
f = 3 / 4;  /*f contient la valeur 0.0 après l'affectation !!! */
```

- ✓ Conversion explicite (cast)
  - Modification forcée du type (si compatible)
  - Syntaxe : (type) expression

```
int x;

x = (int) 3.14;  /*x contient l'entier 3 (valeur tronquée)
                                    après l'affectation */
```


Éléments de base du langage

- Constantes
  - Mot clé `const`
  - Valeur non modifiable

```
const char c = 'A';   /* non modifiable */
char c = 'A';         /* modifiable */
```

  - Caractères spéciaux indiqués par un anti-slash

\n	À la ligne	\t	Tabulation	\0	Caractère nul
\\	Anti-slash	\a	Bip terminal	\"	Guillemets

  - Constantes nommées
    - Ne pas utiliser les constantes "en dur" dans le programme
      - Nommer les constantes en début de programme
    - Directive au préprocesseur `#define NOM_CONSTANTE valeur`
      - Facilite la mise à jour des constantes dans un programme
      - Substitution dans toute la suite du programme

```
#define TVA 20
```


Éléments de base du langage

- Constantes
  - 4 types de constantes
    - Constantes entières
      - Représentation en base 10, en base 8 (on fait précéder le nombre de 0) ou en base 16 (on fait précéder le nombre de 0x ou 0X)
    - Constantes réelles
      - Notation décimale (3.141) ou "scientifique" (0.3141e+1 ou 0.3141E+1)
    - Constantes caractères
      - Caractères imprimables ('A' ou 'a')
      - Caractères disposant d'une séquence d'échappement ('\n' : saut de ligne, '\t' : tabulation, '\f' : saut de page, '\"' guillemets …)
      - Désignation d'un caractère par son code (\x41 pour 'A', 65 pour 'A' …)
    - Constantes chaînes de caractères
      - Placées entre guillemets ("bonjour", "a", "abc\"cd\"")
      - En mémoire, une constante chaîne de caractères est complétée par '\0'
        - Ne pas confondre le caractère 'A' occupant 1 octet et la chaîne "A" occupant 2 octets

Éléments de base du langage

- ✓ Opérateurs
  - Arithmétiques
    - Unaire
      - - (moins unaire)
        - short som = 1; som = -som;
    - Binaire
      - + (addition)
        - short a, b, s; a = 1; b = 2; s = a + b;
      - - (soustraction)
        - short a, b, s; a = 1; b = 2; s = a - b;
      - * (multiplication)
        - short a, b, s; a = 1; b = 2; s = a * b;
      - / (division)
        - short a, b, s; a = 1; b = 2; s = a / b;
      - % (modulo)
        - short a, b, s; a = 1; b = 2; s = a % b;

        (* reste de la division entière de a par b *)

Éléments de base du langage

- Logiques
  - non : !
  - et : &&
  - ou : ||

a	b	non a	a et b	a ou b
V	V	F	V	V
V	F	F	F	V
F	V	V	F	V
F	F	V	F	F

- Relationnels
  - == opérateur de test d'égalité
  - != opérateur de test d'inégalité
  - <= opérateur d'infériorité ou d'égalité
  - < opérateur d'infériorité
  - > opérateur de supériorité
  - >= opérateur de supériorité ou d'égalité

Éléments de base du langage

- Priorité des opérateurs (décroissante)
  - !, - (unaire)
  - *, /, %
  - +, -
  - <, <=, >=, >
  - ==, !=
  - &&
  - ||
  - Les opérateurs de même priorité sont évalués de la gauche vers la droite
- Exemples
  - L'expression ((2>3) && (a==a)) || (2!=3) renvoie true
  - L'expression (2>3) && ((a==a) || (2!=3)) renvoie false
  - L'expression (2>3) && (a==a) || (2!=3) renvoie true
    - ➔ utilisation des parenthèses pour lever toute ambiguïté

Éléments de base du langage

- Les opérateurs d'incrémentation / de décrémentation
  - ++x et x++ sont toutes deux équivalentes à l'affectation x=x+1
  - --x et x-- sont toutes deux équivalentes à l'affectation x=x-1
- Mais
  - Lorsque ces opérateurs sont utilisés dans des sous-expressions
    - L'opérateur de pré-incrémentation (resp. pré-décrémentation) incrémente (resp. décrémente) la variable avant de l'utiliser dans l'expression englobante
    - L'opérateur de post-incrémentation (resp. post-décrémentation) ne l'incrémente (resp. décrémente) qu'après l'avoir utilisée dans l'expression avec sa valeur originelle

Éléments de base du langage

- Exemple

```
#include <stdio.h>
/* Test des opérateurs d'incrémentation et de décrémentation */

int main ()
{
    int x = 1, y = 0;
    y = ++x;
    printf("x = %d \t y = %d\n", x, y);
    y = x++;
    printf("x = %d \t y = %d\n", x, y);
    return 0;
}
```


Éléments de base du langage

- Les entrées/sorties élémentaires
  - #include <stdio.h>
    - int getchar();
      - Lecture d'un caractère
      - En fin de fichier : retour de EOF
    - putchar (<char>);
      - Affichage d'un caractère
    - printf (format, x1, …, xn);
      - %s : affiche l'argument suivant en tant que chaîne de caractères
      - %d : affiche l'argument suivant en tant qu'entier
      - %c : affiche l'argument suivant en tant que caractère
      - %f : affiche l'argument suivant en tant que réel
      - %% : afficher un %
    - scanf (format, &x1, …, &xn)
      - %d : lecture d'un entier
      - %f : lecture d'un réel
      - …
      - L'opérateur & désigne l'adresse (référence) d'une variable

```
int nb;
printf ("entrer un entier");
scanf ("%d",&nb);
printf ("entier saisi :%d\n",nb);
```


Éléments de base du langage

- 3 structures de contrôle
  - Séquence
    - Exprime l'enchaînement inconditionnel et séquentiel d'un ensemble d'actions (élémentaires ou composées)

Séquence

```
//action 1;
//action 2;
//…
//action n;
```

```
//…
int nb;
printf("Saisir la valeur de nb :");
scanf ("%d", &nb);
printf ("La valeur de nb est egale a %d\n", nb);
```


Éléments de base du langage

- Sélection
  - Exprime une possibilité de choix entre deux actions (élémentaires ou composées) en fonction de la valeur de vérité d'une condition (vrai ou faux)
  - L'instruction "if"

```
if (<expression>)
   <instruction1>;
[else
   <instruction2>;
]
```

```
//…
if (delta < 0)
   printf ("racines imaginaires\n");
else
   if (delta == 0)
      x0 = -b / (2*a);
   else
   {
      x1 = (-b - sqrt(delta)) / (2*a);
      x2 = (-b + sqrt (delta)) / (2*a);
   }
```

Éléments de base du langage

- Sélection
  - L'instruction "switch"

```
switch (<expression-entière>)
{
   case <val1> : <suite_instr1>; break;
   case <val2> : <suite_instr2>; break;
   …
   case <valn> : <suite_instrn>; break;
   default     : <suite_instr>;
}
```

• L'option `default` regroupe les valeurs non spécifiées explicitement
• Le contrôle ne sort pas de l'instruction après traitement d'un choix => utiliser `break`

```
//…
switch (d)
{
   case '1' : printf("jour 1\n");
   case '2' : printf("jour 2\n");
   case '3' : printf("jour 3\n");
   default  : printf ("autre \n");
}
```

Trace
Si d = '2'

Trace
Si d = '2'

Éléments de base du langage

- Répétition
  - Exprime le fait qu'une action (élémentaire ou composée) sera répétée tant que la valeur de vérité d'une condition sera vraie. La valeur de vérité de la condition doit être modifiée par l'action, sinon la répétition pourra être infinie
  - La boucle "tant que" : "while"

```
while (<expression>)
   <instruction>;
```

```
int i = 1;
while (i<=3)
{
   printf ("Bonjour !!!\n");
   i = i + 1;
}
```

  - La boucle "do"

```
do
   <instruction>;
while (<expression>);
```

```
int i = 1;
do
{
   printf ("Bonjour !!!\n");
   i = i + 1;
} while (i<=3);
```


Éléments de base du langage

  - La boucle "pour" : "for"

```
for (<inst-initiale>; <test-boucle>; <inst-incrémentation>)
   <instruction>;
```

```
int i;
for (i=1; i<=3; i=i+1)
   printf ("Bonjour !!!\n");
```

  - La boucle "for" est équivalente au schéma ci-dessous qui utilise la boucle "while"

```
<inst-initiale>;
while (< test-boucle >)
{
   <instruction>;
   <inst-incrémentation>;
}
```

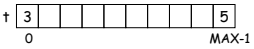

Exercices

✓ Exercice
» Écrire un programme C permettant de lire un nombre puis d'afficher à l'écran un message indiquant si ce nombre entier est pair ou pas
  - On utilisera l'opérateur `%`
    - `a % b` retourne le reste de la division entière de l'entier a par l'entier b (5 % 2 retourne 1)

✓ Exercice
» Écrire un programme C permettant de calculer la somme des 10 premiers nombres entiers en utilisant les 3 structures de contrôle répétitives puis d'afficher le résultat à l'écran

Tableaux

✓ Tableau à une dimension
» Permet de stocker plusieurs éléments, TOUS étant de même type
» Exemple



» Syntaxe

```
type_des_elements nomTableau[taille];
```

  - Les éléments d'un tableau sont désignés par des indices (calculables) qui sont des entiers compris entre 0 et la longueur du tableau -1

» Accès à une case particulière du tableau

```
nomTableau[indice];
```

» Pour remplir un tableau
  - Saisir une à une chaque valeur

» Pour afficher le contenu d'un tableau
  - Afficher une à une chaque valeur
    - ➔ Utilisation de la répétition

```
#define MAX 10
//…
int t[MAX];
t[0] = 3;
t[MAX-1] = 5;
```


Exercice

» Exercice
  - Ecrire un programme C permettant de
    - remplir un tableau d'entiers comportant 5 cases; ces entiers seront saisis par l'utilisateur
    - afficher le contenu de ce tableau d'entiers
    - échanger le contenu des cases d'indice 1 et 3 du tableau
    - afficher le contenu de ce tableau d'entiers modifié

» Exercice
  - Écrire un programme C qui permet de
    - lire N entiers saisis au clavier (N sera saisi par l'utilisateur) et de les stocker dans un tableau puis
    - de calculer la somme des N entiers saisis et rechercher le minimum parmi ces valeurs