ISEN
ALL IS DIGITAL!
LILLE

yncréa

CIR²

Rappel

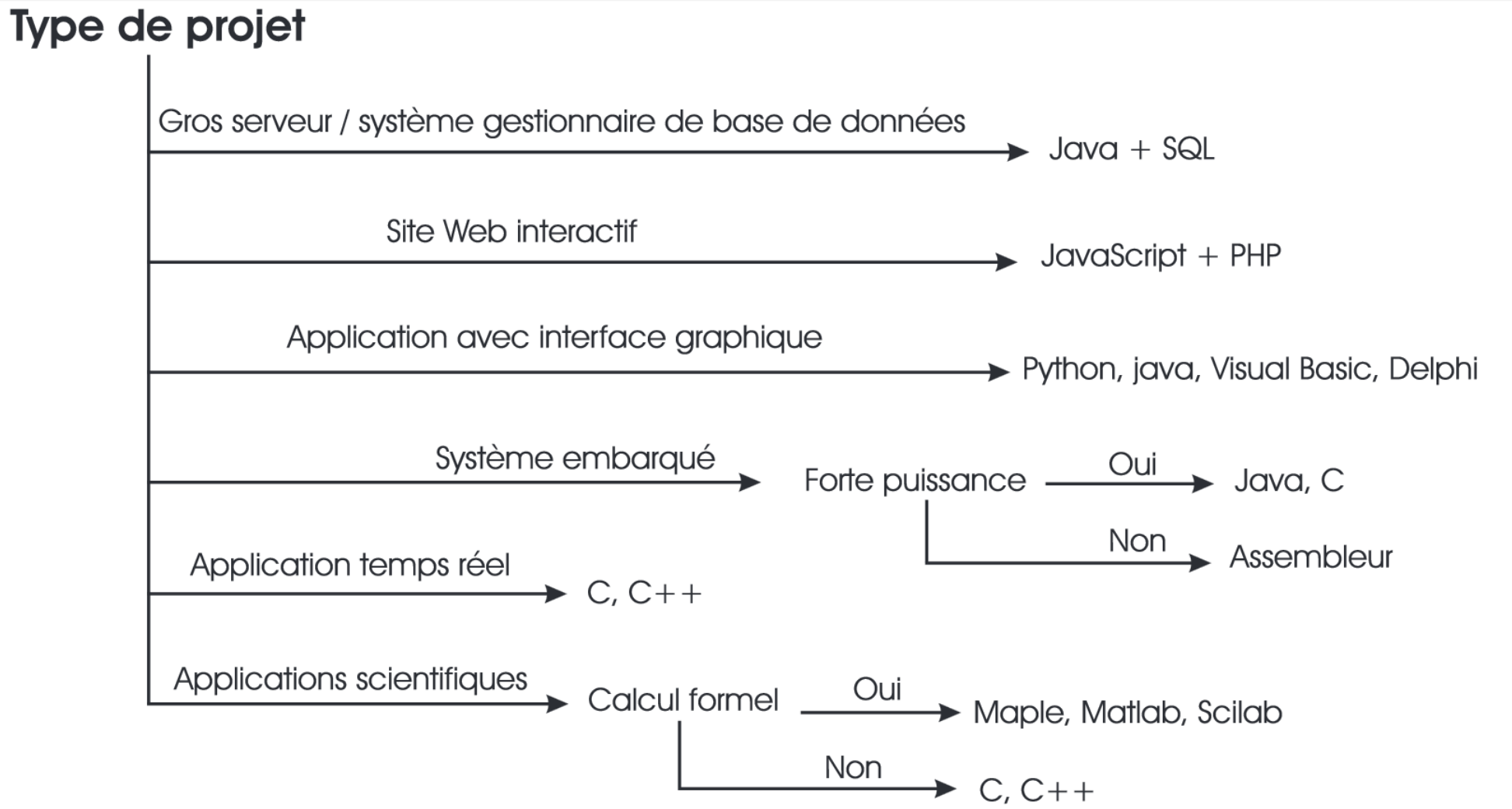
Programmation Langage C

Nacim IHADDADENE

Introduction

- Inventé par **Dennis Richie & Ken Thompson** (Bell Labs) pour développer le systèmes d'exploitation UNIX.
- Le langage C a fait l'objet d'une norme ANSI à partir de 1989 (C89, C90, C ANSI, C99, C11).
- Langage de bas niveau.
- Langage structuré / impératif / procédural.
- Compilateurs gratuits sur Linux: GNU gcc, Intel icc.
- Les autres langages tels que C++, Java, C#, Objective C, JavaScript, Perl et PHP, etc. reprennent des aspects du langage C.
- Quel IDE ?
  Liste des environnements de développement C/C++

Choix d'un langage de programmation



Plan

- La structure d’un programme en C
- Les déclarations de variables
- Opérateurs
- Instructions itératives
- Instructions conditionnelles
- Fonctions
- Bibliothèques standards

Structure d'un programme

- Les programmes sont structurés en fichiers entêtes (.h) et fichiers sources (.c)
- L'entrée du programme est la fonction obligatoire **`main()`**
- Les commentaires entre /* et */ (ou sur une ligne après **//**)
  Il permettent de documenter le code ou d'en désactiver certaines parties
- Un programme en C est constitué de :

  mots clés, identificateurs,
  constantes, chaînes de caractères,
  opérateurs, séparateurs,
  commentaires, directives de compilation.

Exemple de structure d'un programme

Fichier.c

```
[ Directives de compilation ]
[ Déclaration de variables globales ou externes ]
[ Déclaration ou défitinion de fonctions secondaires ]
main ()
{
    [ déclarations de variables locales (internes) ]
    [ instructions ]
}
```

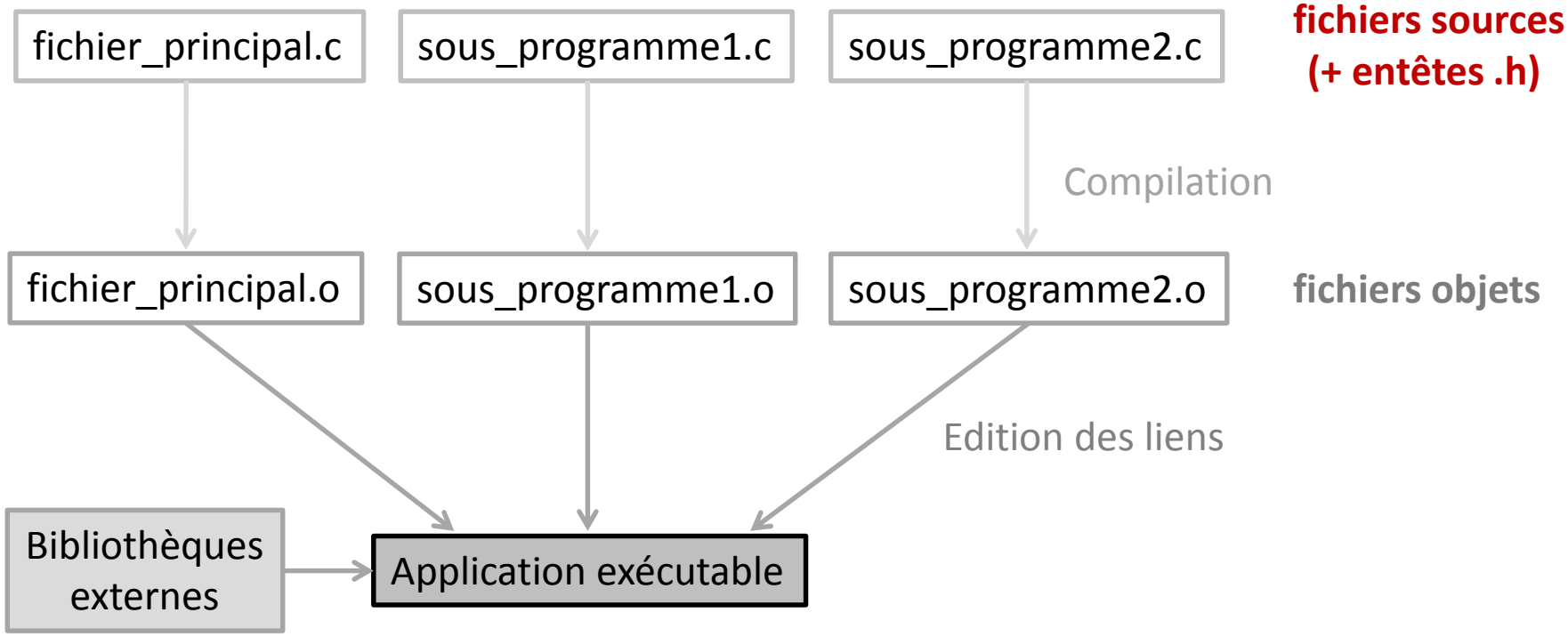
Attention à la portée des variables!

Exemple : Hello world!

```
/* Hello World program */
#include<stdio.h>
main()
{
      printf("Hello World");
}
```

```
/* Hello World program */
#include<stdio.h>
main()
{
      printf("Hello World");
}
```

Génération d'une application



- **Commandes gcc**

```
gcc -c nom_programme.c
gcc fichier1.o … fichierN.o -o nom_executable
```

Variables : identificateurs

- Noms donnés aux variables et aux fonctions.
- Une suite de lettres (majuscules ou miniscules, non accentuées) ou de chiffres.
  - Commencent obligatoirement par une lettre.
  - Le caractère _ (souligné) est considéré comme une lettre.
  - Respect de la casse.
- Les mots clés (mots réservés) ne peuvent pas être utilisés comme identificateurs :

auto break case char const continue default do double else enum extern	float for goto if inline (since C99) int long register restrict (since C99) return short	signed sizeof static struct switch typedef union unsigned void volatile while	_Alignas (since C11) _Alignof (since C11) _Atomic (since C11) _Bool (since C99) _Complex (since C99) _Generic (since C11) _Imaginary (since C99) _Noreturn (since C11) _Static_assert (since C11) _Thread_local (since C11)

Variables : Types de base

Type	Signification	Exemples	mémoire	dérivés
char	Caractère unique	'a' 'A' 'z' 'Z' '\n' 'a' 'A' 'z' 'Z' '\n'	1 octet	signed, unsigned
int	Nombre entier	0 1 -1 2017 -320000	2 ou 4 octets	Short, long, signed, unsigned
float	Nombre réel simple	0.0 1.0 3.14 5.32 -1.23	4 octets	
double	Nombre réel double précision	0.0 1.0E–10 1.0 -1.34567896	8 octets	long

- Utiliser la librairie <limits.h> pour avoir les valeurs minimales et maximales du compilateur pour les différents types (ex. SCHAR_MIN vaut -128).

Variables : Tableaux

- Déclaration:

  type identificateur[taille];

  ex: un tableau « tab » de 100 entiers: int tab[100];

- Accès : opérateur [] :

  Ex : tab[19] = 0; tab[i-1] = a;

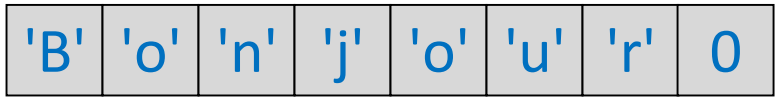
- Les indices vont de 0 à Taille-1.
- Les tableaux peuvent être initialisés:

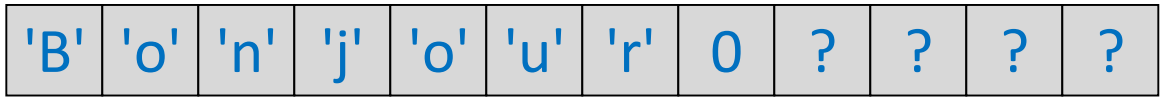
  Ex : int A[5]={1,2,3,5,7};

Les chaînes de caractères

- Il n'existe de pas de type prédéfini pour les chaînes de caractères.
  Elles sont représentées par des tableaux de caractères.
- Convention : Chaînes à zéro terminal (\0 , code ascii 0)
- Exemples :

```
char t[8] = "Bonjour";
char t[12]= "Bonjour";
```





Variables : Structures

- Syntaxe de définition:

```
struct nom_strcture {
    type1 champ1;
    type2 champ2;
    type3 champ3;
    …
};
```

- Déclaration : `struct nom_strcture nom_variable;`
- Accès par l'opérateur : `.`

```
nom_variable.champi;
```

Opérateurs (1/3)

- Opérateur d'affectation : **=**
- Opérateurs arithmétiques : **+**, **-**, *****, **/**, **%**
- Opérateurs relationnels : **>**, **>=**, **<**, **<=**, **==**, **!=**
- Opérateurs logiques
  - Booléens : **&&**, **||**, **!**
  - Bit à Bit : **&**, **|**, **^**, **~**, **<<**, **>>**
- Evaluation conditionnelle : **?** **:**

Opérateurs (2/3)

- Opérateurs d'affectation composée
  +=, **-=**, ***=**, **/=**, **&=**, **^=**, **|=**, **<<=**, **>>=**

- Opérateurs d'incrémentation ou de décrémentation
  ++, **--** (en suffixes ou en préfixe)

Opérateurs (3/3)

- Opérateurs d'accès mémoire :
  - Adresse : **&**
  - Indirection : *****
  - Elément d'un tableau : **[ ]**
  - Membre d'une structure : **.**
  - Membre d'une structure pointée : **->**
- Conversion de type (cast) : **( )**
- Taille en octets : **sizeof( )**

Instructions conditionnelles

- **if else**

```
if (expression_1)
    Bloc_Instructions_1
else if (expression_2)
    Bloc_Instruction_2
...
```

  - Le **else** est facultatif

```
if (expression)
    instruction
```

Instructions conditionnelles

- **Switch**

```
switch (expression)
 { case constante1 :
        instructions1;
        break;
   case constante2 :
        instructions2;
        break;
   ...
   default :
        instructions;
        break;
}
```

Instructions itératives

- La boucle **while :**

```
while (expression)
    instructions;
```

- La boucle **do while :**

```
do
    instructions;
while (expression);
```

- La boucle **for** :

```
for (expr1; condition; expr2)
    instructions;
```

- Branchement non conditionnel: **break**, **continue**

Fonctions en C

- Déclaration:

```
Type_de_retour nom_fonction(  type_1 param_1, …, type_n param_n);
```

- Définition:

```
Type_de_retour nom_fonction(  type_1 param_1, …, type_n param_n)
{
    /* code de la fonction */
    return var_retour;
}
```

- Appel de la fonction:

```
nom_fonction(liste_parametres);
```

Bibliothèques standards

- Utilisation : insertion en début du programme de la directive

```
#include <_____.h>
```

- **<stdio.h>** : pour les entrées/sorties de base

  `printf`, `scanf`, `fopen`, `getchar`, …

- **<stdlib.h>** : fonctions générales

  `malloc`, `rand`, `system`, `atoi`, …

- **<math.h>** : fonctions mathématiques de base

  `sqrt`, `pow`, `cos`, `tan`, `ceil`, `floor`, …

- **<string.h>** : manipulation des chaînes de caractères

  `strcmp`, `strlen`, `strcat`, …

- Etc. (voir la référence du compilateur utilisé)

Conclusion

- Le cours ne mentionne pas toutes les possibilités du langage C et renvoie vers des ressources plus spécialisées.
- Ce cours n'est qu'un rappel des principales caractéristiques du langage C avant de parler du langage C++.
- De nombreux tutoriels existent sur internet, pour ceux qui désirent aller plus loin que ce petit rappel.
- http://fr.cppreference.com/w/c
- https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html