COURS PROGRAMMATION : UN PEU PLUS DE C

CHARLES 'ARYS' YAICHE

RÉCAP DU COURS PRÉCÉDENT

- Ce qu'étais un programme, ainsi qu'un langage de programmation
- Les étapes de compilation
- Introduction au C

OBJECTIF DU COURS

- La mémoire
- Directive de préprocesseur, include, define, macro
- Les fonctions en C
- Les boucles en C

ANALYSE D'UN PROGRAMME EN C

Directive de préprocesseur

Fonction main

- Déclaration de la fonction main
- Appel de la procédure printf
- Retour de la fonction main

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

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

DIRECTIVE DE PRÉPROCESSEUR

Le préprocesseur est un programme exécuté lors de la première phase de la compilation. Il effectue des modifications textuelles sur le fichier source à partir de directives. Les différentes directives au préprocesseur, introduites par le caractère #, ont pour but :

- l'incorporation de fichiers source (#include)
- la définition de constantes symboliques et de macros (#define)

DIRECTIVE DE PRÉPROCESSEUR : INCLUDE

- Elle permet d'incorporer dans le fichier source le texte figurant dans un autre fichier. Ce dernier peut être un fichier en-tête de la librairie standard (stdio.h, math.h,...) ou n'importe quel autre fichier. La directive #include possède deux syntaxes voisines :
  - #include <nom-de-fichier>
    - recherche le fichier mentionné dans un ou plusieurs répertoires systèmes définis par l'implémentation (par exemple, /usr/include/) ;
  - #include "nom-de-fichier"
    - recherche le fichier dans le répertoire courant (celui où se trouve le fichier source). On peut spécifier d'autres répertoires à l'aide de l'option -I du compilateur.
- La première syntaxe est généralement utilisée pour les fichiers en-tête de la librairie standard, tandis que la seconde est plutôt destinée aux fichiers créés par l'utilisateur.

DIRECTIVE DE PRÉPROCESSEUR : DEFINE

- La directive #define permet de définir :
  - des constantes symboliques
  - des macros avec paramètres

La directive
#define nom reste-de-la-ligne

demande au préprocesseur de substituer toute occurrence de nom par la chaîne de caractères reste-de-la-ligne dans la suite du fichier source. Son utilité principale est de donner un nom parlant à une constante, qui pourra être aisément modifiée. Par exemple :

#define NB_LIGNES 10
#define NB_COLONNES 33
#define TAILLE_MATRICE NB_LIGNES * NB_COLONNES

Il n'y a toutefois aucune contrainte sur la chaîne de caractères reste-de-la-ligne. On peut écrire

#define BEGIN {
#define END }

```c
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0

int main() {
    printf("Hello, World!\n");
    int a = TRUE;
    if (!a){
        printf("false \n");
    }
    return 0;
}
```

LES FONCTIONS ET PROCÉDURES

- Syntaxe de déclaration de fonction C
- Pour déclarer une procédure (fonction sans valeur de retour il faut utiliser le mot clefs void

Les paramètres sont déclaré tel que :

- nomFonction(type paramètre1, type paramètre2)

```c
void test(int a, char *b){
    // instructions
}

int sum(int a, int b){
    return a + b;
}

int f(int x, int a, int b){
    return a * x + b;
}
```

DÉCLARATION VS DÉFINITION

Déclaration

- La déclaration (ou prototypage) permet de donner les caractéristiques (type de retour, identifiant et paramètres)

Définition

- Expression qui code la fonctions (ou la procédure)

```
//déclaration
int sum(int a, int b);

// définiton
int sum(int a, int b){
    return a + b;
}
```

LES BOUCLES : WHILE

```c
int ii = 0;
while (ii < 10) {
    printf("%d, ", ii++);
} // => prints "0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "
```

LES BOUCLES : DO … WHILE

```c
int kk = 0;
do {
    printf("%d, ", kk);
} while (++kk < 10);
// => prints "0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "
```

LES BOUCLES : FOR

```
// For loops too
int jj;
for (jj=0; jj < 10; jj++) {
    printf("%d, ", jj);
} // => prints "0, 1, 2, 3, 4, 5, 6, 7, 8, 9, "
```

DIFFICULTÉ DES EXO

- Facile
- Moyen
- difficile