CRO: C langage and programming Roots

tanguy.risset@insa-lyon.fr
Lab CITI, INSA de Lyon
Version du January 28, 2019

Tanguy Risset

January 28, 2019

Table of Contents

Types construits

- À partir des types prédéfinis du C (caractères, entiers, flottants), on peut créer de nouveaux types, appelés types contruits, qui permettent de représenter des ensembles de données organisées.
- Les tableaux
- Les structures
- Les unions
- Les énumérations
- Les constructeurs de type

Les tableaux

- Un tableau est un ensemble fini d'éléments de même type, stockés en mémoire à des adresses contiguës.
- La déclaration d'un tableau `tab` de 10 entiers se fait de la façon suivante : `int tab[10];`
- On accède au troisième élément du tableau `tab` par l'expression `tab[2]`. Les tableaux en C *commencent toujours à 0*.
- On peut définir des tableaux multidimensionnels. Exemple: matrice `M` d'entiers de 10 lignes 5 colonnes `int M[10][5]`
- Important: un tableau en C peut être vu comme un pointeur sur le premier élément du tableau:
  `int *tab;` $\Leftrightarrow$ `int tab[];`
- Mais il faut savoir qu'en C un tableau est une constante

Tableaux: exemple

```
char tab[10] = {'e','x','e','m','p','l','e','\0'};
main()
{
  int i;
  for (i = 0; i < 10; i++)
    printf("tab[%d] = %c\n",i,tab[i]);
}
```

```
tab[0] = e
tab[1] = x
tab[2] = e
tab[3] = m
tab[4] = p
tab[5] = l
tab[6] = e
tab[7] =
```


Tableaux: exemple

```
#define M 2
#define N 3
int tab[M][N] = {{1, 2, 3}, {4, 5, 6}};

main()
{
  int i, j;
  for (i = 0 ; i < M; i++)
    {
      for (j = 0; j < N; j++)
        printf("tab[%d][%d]=%d\n",i,j,tab[i][j]);
    }
}
```

```
tab[0][0]=1
tab[0][1]=2
tab[0][2]=3
tab[1][0]=4
tab[1][1]=5
tab[1][2]=6
```

Retour sur les types: les chaînes de caractères

- En C une chaine de caractères n'est pas un type de base: c'est un tableau de caractères terminé par le caractère spécial '\0' (ou NULL: octet valant 0).
- On peut la noter entre double cote:
  `char chaine[10]="bonjour"`
- Où comme un tableau de caractère (jamais utilisé):
  `char chaine[10]={'b','o','n','j','o','u','r','\0'}`
- Comme un tableau en C est assimilé à un pointeur sur le début du tableau on trouvera souvent des chaines de caractères déclarées comme:
  `char *chaine;`
- Pour l'instant on preferera la déclaration statique (tableau). Attention il faut une case de plus que le nombre de lettres (pour '\0').

Les structures

- Une structure est une suite finie d'objets de types différents.
- Contrairement aux tableaux, les différents éléments d'une structure n'occupent pas nécessairement des zones contiguës en mémoire.
- Chaque élément de la structure, appelé membre ou champ, est désigné par un identificateur.
- Deux manières équivalentes de définir une variable z complexe:

```
struct complexe
{
  double reelle;
  double imaginaire;
} z;
```

```
struct complexe
{
  double reelle;
  double imaginaire;
};
struct complexe z;
```

```
norme=sqrt(z.reelle*z.reelle+z.imaginaire*z.imaginaire);
```

Les structures de contrôle de flot

- Les objets que nous avons rencontrés permettent de faire des calculs, pas de *contrôler quels calculs* sont faits
- Les structures de contrôles sont de deux types:
  - Les boucles
  - Les instructions de branchement

Table of Contents

Les boucles

- Boucles `while`:

```
while (expression )
  instruction
```

```
i = 1;
while (i < 10)
  {
    printf(" i = %d \n",i);
    i++;
  }
```

- Boucles `for`

```
for (expr 1 ;expr 2 ;expr 3)
  instruction
```

$\Leftrightarrow$

```
expr 1;
 while (expr 2 )
   {instruction
    expr 3;
   }
```


Les instructions de branchement

- Branchement conditionnel

```
if (expression-1 )
   instruction-1
else instruction-2
```

ou

```
if (expression-1 )
   instruction-1
```

- Branchement multiple `switch`:

```
switch (expression )
  {case constante-1:
    liste d'instructions 1
    break;
  case constante-2:
    liste d'instructions 2
    break;
  case constante-3:
    liste d'instructions 3
    break;
  default:
    liste d'instructions defaut
    break;
  }
```

- Branchement non conditionnel: `break`, `continue`
- Branchement à ne pas utiliser: `goto`

Table of Contents

Fichiers

- Les accès à un fichier se font par l'intermédiaire d'un tampon (buffer) qui permet de réduire le nombre d'accès aux périphériques (disque...).
- Pour pouvoir manipuler un fichier, un programme a besoin d'un certain nombre d'informations : l'adresse du tampon, position dans le fichier, mode d'accès (lecture ou écriture) ...
- Ces informations sont rassemblées dans une structure dont le type, `FILE *`, est défini dans `stdio.h`. Un objet de type `FILE *` est un stream (flot).
- Avant de lire ou d'écrire dans un fichier, on l'*ouvre* par la commande `fich=fopen("nom-de-fichier","r")`. Cette fonction dialogue avec le système d'exploitation et initialise un stream `fich`, qui sera ensuite utilisé lors de l'écriture ou de la lecture.
- Après les traitements, on *ferme* le fichier grâce à la fonction `fclose(fich)`.

Fichiers

- on peut ouvrir un fichier sous plusieurs modes: lecture ("r"), écriture au début ("w"), écriture à la fin ("a"). `fopen` retourne 0 en cas d'echec.
- Exemple:

```
FILE *fich;
fich=fopen("./monFichier.txt","r");
if (!fich) fprintf(stderr,"Erreur d'ouverture : %s\n","./monFichier.txt");
```

- Un objet de type `FILE` est quelquefois appelé un déscripteur de fichier, c'est un entier désignant quel est le fichier manipulé.
- Trois descripteurs de fichier peuvent être utilisés sans qu'il soit nécessaire de les ouvrir (à condition d'utiliser `stdio.h`):
  - `stdin` (standard input) : unité d'entrée (par défaut, le clavier, valeur du descripteur: 1) ;
  - `stdout` (standard output) : unité de sortie (par défaut, l'écran, valeur du descripteur: 0) ;
  - `stderr` (standard error) : unité d'affichage des messages d'erreur (par défaut, l'écran, valeur du descripteur: 2).
- Il est fortement conseillé d'afficher systématiquement les messages d'erreur sur stderr afin que ces messages apparaissent à l'écran même lorsque la sortie standard est redirigée.

Autres fonction de lecture/ecriture

- Entrée/sorties de caractères
  - `int fgetc(FILE* flot);`
  - `int fputc(int caractere, FILE *flot)`
- Entrée/sorties de chaînes de caractères
  - `char *fgets(char *chaine, int taille, FILE* flot);`
  - `int fputs(const char *chaine, FILE *flot)`
- Entrée/sorties binaires
  - `size_t fread(void *pointeur, size_t taille, size_t nombre, FILE *flot);`
  - `size_t fwrite(void *pointeur, size_t taille, size_t nombre, FILE *flot);`
- positionnement dans un fichier
  - `int fseek(FILE *flot, long deplacement, int origine);`
  - trois valeurs possibles pour origine: `SEEK_SET` (égale à 0) : début du fichier ,`SEEK_CUR` : position courante, `SEEK_END` (égale à 2) : fin du fichier.

fgetc, fputc: exemple

```
##include <stdio.h>
##include <stdlib.h>
##define ENTREE "entree.txt"
##define SORTIE "sortie.txt"
int main(void)
{FILE *f_in, *f_out;
  int c;

  if ((f_in = fopen(ENTREE,"r")) == NULL)
    {
      fprintf(stderr, "\nErreur: Impossible de lire le fichier %s\n",ENTREE);
      return(EXIT_FAILURE);
    }
  if ((f_out = fopen(SORTIE,"w")) == NULL)
    {
      fprintf(stderr, "\nErreur: Impossible d'ecrire dans le fichier %s\n",
      SORTIE);
      return(EXIT_FAILURE);
    }
  while ((c = fgetc(f_in)) != EOF)
    fputc(c, f_out);
  fclose(f_in);
  fclose(f_out);
  return(EXIT_SUCCESS);
}
```