C/C++: Structures, Fichiers.

Vincent Gaudillière

Polytech Nancy

2017 / 2018

Les structures.

Une structure:

- peut être vue comme un type de variable personnalisé,
- est un assemblage de variables de types déjà connus.

Les structures.

Une structure:

- peut être vue comme un type de variable personnalisé,
- est un assemblage de variables de types déjà connus.

Mode d'emploi:

1. Définition de la structure *MaStructure*,
2. Déclaration d'une variable *maVariable* de type *MaStructure*,
3. Accès (lecture, et affectation de valeurs) aux éléments de *maVariable*.

Définition d'une structure (fichier *.h).

```
struct MaStructure
{
    int unEntier;
    double unTableauDeDoubles[10];
    char unChar;
};
```

Définition d'une structure (fichier *.h).

```
struct MaStructure
{
    int unEntier;
    double unTableauDeDoubles[10];
    char unChar;
};
```

```
typedef struct
{
    int unEntier;
    double unTableauDeDoubles[10];
    char unChar;
}MaStructure;
```

Déclaration d'une variable de type structure (cas 1).

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

struct Point // définition SANS "typedef".
{
    double x; // abscisse
    double y; // ordonnée
};

int main(){
    struct Point P; // déclaration: syntaxe n°1.
    return 0;
}
```

Déclaration d'une variable de type structure (cas 2).

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

typedef struct // définition AVEC "typedef"
{
    double x; // abscisse
    double y; // ordonnée
}Point;

int main(){
    Point P; // déclaration: syntaxe n°2.
    return 0;
}
```

Les structures: accès aux valeurs.

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

typedef struct
{
    double x;
    double y;
}Point;

int main(){
    Point P = {0,2}; // initialisation (pas obligatoire).
    P.x = 1; // affectation
    printf("Ordonnée: %lf\n",P.y); // lecture
    return 0;
}
```

Structures et pointeurs.

```
Point P = {1,2};
Point * p_P = &P;

printf("Abscisse: %lf\n",P.x); // 1
printf("Abscisse: %lf\n",(*p_P).x); // 1

printf("Abscisse: %lf\n",p_P->x); // 1
printf("Abscisse: %lf\n",(&P)->x); // 1
```

Tableaux de structures.

```
int main()
{
    Point tab[10];
    int i;

    for(i=0;i<10;i++)
    {
        tab[i].x = 2*i; // (A) équivalent à (B) et (C)
        (*(tab+i)).x = 2*i; // (B)
        (tab+i)->x = 2*i; // (C)
    }
}
```

Lire et écrire dans un fichier.

- Les variables créées dans un programme n'existent que pendant le temps d'exécution de ce programme.

Lire et écrire dans un fichier.

- Les variables créées dans un programme n'existent que pendant le temps d'exécution de ce programme.
- Or la plupart des programmes ont besoin d'enregistrer certaines données, pour les retrouver lors de la prochaine exécution par exemple (éditeur de texte, jeu vidéo, etc...).

Lire et écrire dans un fichier.

- Les variables créées dans un programme n'existent que pendant le temps d'exécution de ce programme.
- Or la plupart des programmes ont besoin d'enregistrer certaines données, pour les retrouver lors de la prochaine exécution par exemple (éditeur de texte, jeu vidéo, etc...).
- Le langage C (comme de nombreux autres langages) permet ainsi d'écrire et de lire dans des fichiers.

Ouvrir et fermer un fichier.

```c
int main(){
    FILE * f = NULL; // toujours initialiser un pointeur!
    f = fopen("fichier.txt","r"); // [r]ead ou [w]rite
    if(f!=NULL) // fichier ouvert
    {
        // ...
        fclose(f); // fermer le fichier!
    }
    else
    {
        printf("Impossible d'ouvrir le fichier.\n");
    }
}
```

Lire dans un fichier (fscanf). (1/2)

`fichier.txt` (1 ligne): [1 3 15]

```c
FILE * f = NULL;
f = fopen("fichier.txt","r"); // ouverture en mode lecture
if(f!=NULL)
{
    int a,b,c;
    fscanf(f,"%d %d %d",&a,&b,&c);
    printf("%d\n",a); // 1
    printf("%d\n",b); // 3
    printf("%d\n",c); // 15
    fclose(f); // fermer le fichier!
}
```

Lire dans un fichier (fscanf). (2/2)

`fichier.txt`: 10 lignes selon le format précédent.

```
FILE * f = NULL;
f = fopen("fichier.txt","r"); // ouverture en mode lecture
if(f!=NULL)
{
    int a[10],b[10],c[10];
    int i;
    for(i=0;i<10;i++)
    {
        fscanf(f,"%d %d %d",&a[i],&b[i],&c[i]);
    }
    fclose(f); // fermer le fichier!
}
```

Ecrire dans un fichier (fprintf). (1/2)

`fichier.txt`: déjà existant ou non.

```
FILE * f = NULL;
f = fopen("fichier.txt","w");// ouverture en mode écriture
if(f!=NULL)
{
    int a=2,b=5,c=10;
    fprintf(f,"%d %d %d",a,b,c);
    fclose(f); // fermer le fichier!
}
```

`fichier.txt` (1 ligne): [2 5 10]

Ecrire dans un fichier (fprintf). (2/2)

`fichier.txt`: déjà existant ou non.

```
FILE * f = NULL;
f = fopen("fichier.txt","w");// ouverture en mode écriture
if(f!=NULL)
{
    int i,a[3]={1,2,3};
    for(i=0;i<3;i++)
    {
        fprintf(f,"%d\n",a[i]);
    }
    fclose(f); // fermer le fichier!
}
```

`fichier.txt` (3 lignes): [1][2][3]