> C <

Structures et types

Dorian.H Mekni

02 | 03 | 2020

Structures

- Elle se declare <struct>
- le nom qu’on lui donne s’écrit avec une lettre majuscule: Player
- Elle s’ouvre et se ferme avec des bracelets {} puis à sa fermeture un “;” y est ajouté

->

struct

-> La struct est tout simplement un ensemble de données rassemblées sous une dénomination.

1er désign de Struct

```c
// 3 ways to define a struct in C
#include <stdio.h>
// 1

struct Player
{
    signed char username[300];
    int numberOfLifes;
    int magicPowers;
};

int main(void)
{
    struct Player p1 = {"Dorian", 99, 99};

    printf("Name of player : %s\n", p1.username);
    printf("Number of lifes : %d | Number of magic tricks : %d\n", p1.numberOfLifes, p1.magicPowers);
    return 0;
}
```

```
Name of player : Dorian
Number of lifes : 99 | Number of magic tricks : 99
Program ended with exit code: 0
```

2eme désign de struct(2)

s utilisons le typedef afin de minimaliser nos lignes de code au sein de notre fichier source. Nous nommons donc la struct par un alias.

2eme désign de Struct

```c
// 3 ways to define a struct in C
#include <stdio.h>
// 2

struct Player
{
    signed char username[300];
    int numberOfLifes;
    int magicPowers;
};

// Replace "struct Player" by "Player"
typedef struct Player Player;

int main(void)
{
    Player p1 = {"Dorian", 99, 99};

    printf("Name of player : %s\n", p1.username);
    printf("Number of lifes : %d | Number of magic tricks : %d\n", p1.numberOfLifes, p1.magicPowers);
    return 0;
}
```

```
Name of player : Dorian
Number of lifes : 99 | Number of magic tricks : 99
Program ended with exit code: 0
```

3eme désign de Struct(3)

- Le typedef va remplacer l'ensemble des données rassemblées dans la struct
- Gamer ici en l'occurrence

3eme désign de Struct

```c
// 3 ways to define a struct in C
#include <stdio.h>
// 3

typedef struct Player
{
    signed char username[300];
    int numberOfLifes;
    int magicPowers;
} Gamer;


int main(void)
{
    Gamer p1 = {"Dorian", 99, 99};

    printf("Name of player : %s\n", p1.username);
    printf("Number of lifes : %d | Number of magic tricks : %d\n", p1.numberOfLifes, p1.magicPowers);
    return 0;
}
```

```
Name of player : Dorian
Number of lifes : 99 | Number of magic tricks : 99
Program ended with exit code: 0
```

Modification de Valeur

**avons une chaîne de caractère et que nous voulons modifier sa valeur , il
r strcpy() de la biblio #include<string.h> afin de modifier sa valeur. Sans
lisation de pointeurs, le code reste compilable mais non optimal**

strcpy()

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

typedef struct Player
{
    signed char username[400];
    int numberOfLifes;
    int magicPowers;
} Gamer;


int main(void)
{
    Gamer p1 = {"Dorian", 99, 99};

    printf("Name of player : %s\n", p1.username);
    printf("Number of lifes : %d | Number of magic tricks : %d\n", p1.numberOfLifes, p1.magicPowers);

    strcpy(p1.username, "Hedy");
    p1.numberOfLifes -= 50;
    p1.magicPowers -=10;

    printf("Name of player : %s\n", p1.username);
    printf("Number of lifes : %d | Number of magic tricks : %d\n", p1.numberOfLifes, p1.magicPowers);

    return 0;
}
```

Passing 'signed char [400]' to parameter of type 'char *' converts between pointers to integer types with different sign

```
Name of player : Dorian
Number of lifes : 99 | Number of magic tricks : 99
Name of player : Hedy
Number of lifes : 49 | Number of magic tricks : 89
Program ended with exit code: 0
```

Pointeurs -> fonction

On intègre les pointeurs via une fonction afin qu'il s'applique à notre code utilisateur, ou bien ici en l'occurrence Gamer // pour nommer les pointeurs, (p*) revient à l'écriture p->

Pointeurs -> fonction en action

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

typedef struct Gamer
{
    char username[300];
    int numberOfLifes;
    int magicPowers;
} Gamer;

/*-------------------------------------------------------------*/

void create_gamer(Gamer *p)
{

    strcpy(p->username , "Hedy"); // (*p).something = Y == p->something = Y
    p->numberOfLifes = 100; // (*p) == p->
    p->magicPowers = 70;
}

/*-------------------------------------------------------------*/

int main(void)
{
    Gamer p1 = {"", 0, 0};

    create_gamer(&p1);

    printf("Name of player : %s\n", p1.username);
    printf("Number of lifes : %d | Number of magic tricks : %d\n", p1.numberOfLifes, p1.magicPowers);

    return 0;
}
```

```
Name of player : Hedy
Number of lifes : 100 | Number of magic tricks : 70
Program ended with exit code: 0
```

L’intégration de tableau

Il est tout à fait possible de créer un tableau de gamers au sein de notre code sourc orienté jeu vidéo en indiquant pour chaque nouveau gamer une valeur numérique a sein de l’index de notre tableau qui commence par 0:

```c
/*-------------------------------------------------------*/

int main(void)
{
    Gamer p1 = {"", 0, 0};

    Gamer players_tableau[6];

    players_tableau[0].numberOfLifes = 25;
    players_tableau[1].numberOfLifes = 18;


    create_gamer(&p1);

    printf("Name of player : %s\n", p1.username);
    printf("Number of lifes : %d | Number of magic tricks : %d\n", p1.numberOfLifes, p1.magicPowers);

    return 0;
}
```

Pointeurs -> fonction en action

```c
/*-------------------------------------------------------*/

int main(void)
{
    Gamer p1 = {"", 0, 0};

    Gamer players_tableau[6];

    players_tableau[0].numberOfLifes = 25;
    players_tableau[1].numberOfLifes = 18;
    players_tableau[2].numberOfLifes = 18;
    players_tableau[3].numberOfLifes = 18;
    players_tableau[4].numberOfLifes = 18;
    players_tableau[5].numberOfLifes = 18;



    create_gamer(&p1);

    printf("Name of player : %s\n", p1.username);
    printf("Number of lifes : %d | Number of magic tricks : %d\n", p1.numberOfLifes, p1.magicPowers);

    return 0;
}
```

enum

- La struct est tout simplement un regroupement de données.
- L’énumération quant à elle, permet d’énumérer un type de données qui peut être bye struct faisant appel à des constantes à savoir des valeurs possibles.
- Sa premiere donnée a toujours la valeur de 0.

enum en action

```
enum FashionBrands

{

    CD,
    GUCCI,
    PRADA,
    CHANEL

};
```

enum - typedef

```c
#include <stdio.h>

typedef enum FashionBrands

{

    CD,
    GUCCI,
    PRADA,
    CHANEL

} Fashion;
```

enum par exemple

```c
#include <stdio.h>

typedef enum FashionBrands

{

    CD,
    GUCCI,
    PRADA,
    CHANEL

} Fashion;


int main(void)

{

    Fashion brand1 = PRADA;
    return 0;
}

```