Computer Programming 1 Lab

2022-12-22

Mozix Chien

Outline

- Input/Output Format
- Struct
- Exercise 12

Input/Output Format

Input/Output Format

`printf`

- Specifier

```c
/* Signed decimal integer */
printf("%d\n", 455);    // 455
printf("%d\n", +455);   // 455
printf("%d\n", -455);   // -455
printf("%ld\n", 2000000000L);   // 2000000000

/* Unsigned octol integer */
printf("%o\n", 455);    // 707
/* Unsigned decimal integer */
printf("%u\n", 455);    // 455
printf("%u\n", -455);   // 4294966841
/* Unsigned hexadecimal integer */
printf("%x\n", 455);    // 1c7
```

Input/Output Format

`printf`

- Output format - integer

```c
printf("%8d***\n", 123);
printf("%8d***\n", -123);
printf("%-8d***\n", 123);
printf("%-8d***\n", -123);
printf("%8d***\n", 123456789);
printf("%8d***\n", -123456789);
printf("\n");
printf("%d\n%d\n", 64, 64);
printf("%04d\n%04d\n", 64, 64);
```

Input/Output Format

`printf`

- Output format - integer (cont.)

```
      123***
    -123***
123     ***
-123    ***
123456789***
-123456789***

64
64
0064
0064
```

Input/Output Format

- Output format - float

```
printf("%f\n", 3.14159);
printf("%10f\n", 3.14159);
printf("%.2f\n", 3.14159);
printf("%10.2f\n", 3.14159);
```

Output:

```
3.141590
  3.141590
3.14
      3.14
```

Input/Output Format

`sprintf`

Write formatted data to string

```
int sprintf( char* str, const char* format, ...)
```

- `str` : string being processed
- `format` : string format you want
- Retuen value:
  - On success, the total number of characters written is returned.
  - On failure, a negative number (EOF) is returned.

Input/Output Format

`sprintf`

```c
#include <stdio.h>
int main(){
    char buf[50];
    int n;
    int a = 5;
    int b = 3;

    n = sprintf(buf, "%d + %d = %d", a, b, a+b);
    printf("%s\n", buf);
    printf("%d\n", n);
    return 0;
}
```

```
5 + 3 = 8
9
```

Input/Output Format

`scanf`

Precise input formatting can be accomplished with `scanf`

```
scanf(format_control_string, other_arguments);
```

- `format_control_string` describes the formats of the input.
- `ither_arguments` are pointers to variables in which the input will be stored.

Input/Output Format

`scanf`

```
// year, month, and day are "int"
scanf("%d-%d-%d", &year, &month, &day);

// year, month, and day are "int"
// *c can ignore any charactor 1 times
scanf("%d%*c%d%*c%d", &year, &month, &day);

// character is a "char"
scanf("%c\n", &c);

// str is a "char" array, namely string
scanf("%s", str);
```

Input/Output Format

`gets`

```
char *gets(char* str)
```

- Reads a line from stdin and stores it into the string pointed to by str.
- It stops when newline character is read or when the end-of-file is reached, whichever comes first.

Input/Output Format

`puts`

```
int puts(const char *str)
```

- str is the string you want to print out. `puts` will print out str with a '`\n`' at the end.
- Retuen value:
  - On success, return a positive integer.
  - On failure, a negative number (EOF) is returned.

Input/Output Format

```c
#include <stdio.h>

int main () {
   char str[50];

   printf("Enter a string : ");
   gets(str);
   puts("You entered: ");
   puts(str);
}
```

Output:

```
Enter a string : This is a cat.
You entered:
This is a cat.
```

Struct

Struct

Declare

```c
struct person{
    int id;
    int height;
    char name[30];
};
```

Struct

Usage

```
struct person{
    int id;
    int height;
    char name[30];
}john;

struct person cena;
struct person persons[100];
```

Struct

Usage

```
for(int i=0;i<n;i++){
    int id, height;
    char name[30];
    scanf("%d %d %s", &id, &height, name);
    persons[i].id = id;
    persons[i].height = height;
    persons[i].name = name;
}

printf("%d\n", persons[0].id);
printf("%d\n", persons[0].height);
printf("%s\n", persons[0].name);
```

Struct

Typedef

```c
typedef struct person person;

struct person{
    int id;
    int height;
    char name[30];
};

person A, B;
A.id = 1, B.id = 2;
A.height = 183, B.height = 155;
```

Struct

Typedef

```c
typedef struct person person;

struct person{
    int id;
    int height;
    char name[30];
};

person A = {
    .id = 0,
    .height = 155,
    .name = "jassica"
};
```

Struct with sort

Struct with sort

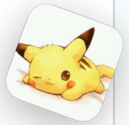
- Using `compare function` to customize your sorting algorithm.

```
struct person{
    int height;
    int weight;
};

int compare(const void *a, const void *b){
    struct person A = *(struct person *)a;
    struct person B = *(struct person *)b;

    // taller people in front
    // or heavier people go first
    if(A.height == B.height){
        return A.weight - B.weight;
    }else{
        return A.height - B.height;
    }
}

int main(void){
    struct person people[100];
    // initialize people ...
    qsort((void *)people, 100, sizeof(people[0]), compare);
}
```



Exercise 12

Exercise 12

Input:

輸入數組神奇寶貝資料
每組資料依序為：
序號、神奇寶貝名、攻擊力、防禦力、血量

Output:

請輸出排序過後的神奇寶貝資料
排序順序為：

```
1. 攻擊力高的排前面
2. 防禦力高的排前面
3. 神奇寶貝名長的排前面
4. 神奇寶貝名字典序
```

注意!!

輸出格式如下：

```
1.序號須為三碼，不足三碼請補零
2.神奇寶貝名長度須為10格，不足10格請留空白
3.攻擊力、防禦力長度須為3格，不足三格請留空白
4.各組資料之間請以一個空白隔開，詳見範例輸出
```

Any Questions?