Computer Programming 1 Lab

2020-12-03

[illegible]

Outline

- Input/Output
- Struct
- Exercise 9
- Assign9 Hint

Input/Output

Input/Output

`printf`

- Specifier

```
/* 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

`printf`

- Output format - integer

```
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

- Output format - integer (cont.)

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

64
64
0064
0064
```

Input/Output

- 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

`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 is returned.

Input/Output

```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;
}
```

Output:

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

Input/Output

`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

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

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

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

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

Input/Output

`gets`

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

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

Input/Output

```c
#include <stdio.h>

int main () {
   char str[50];

   printf("Enter a string : ");
   gets(str);

   printf("You entered: %s", str);

   return(0);
}
```

Output:

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

Struct

Struct

- Structures are collections of related variables under one name.
- **Structures** may contain variables of **many different data types**.
    - **Arrays** contain only elements of **the same data types**.

```
struct student{
    char name[20];
    char gender;
    int age;
    struct student* next;
};
```

```c
struct student stud;
strcpy(stud.name, "Chi-Hung");
stud.gender = 'M';
stud.age = 22;

printf("Name: %s\n", stud.name);
printf("Gender: %c\n", stud.gender);
printf("Age: %d\n", stud.age);
```

Output:

```
Name: Chi-Hung
Gender: M
Age: 22
```

Typedef

- Define the structure first, then use `typedef`.

```
struct student{
    char name[20];
    char gender;
    int age;
    struct student* next;
};
typedef struct student Student;
```

- Use `typedef` when defining the structure.

```
typedef struct student{
    char name[20];
    char gender;
    int age;
    struct student* next;
} Student;
```

Exercise 9

[illegible]

- [illegible]
- [illegible]
- [illegible]The[illegible]the[illegible]
- [illegible]

Get exercise 9 folder by command line

```
oj get_assign ex9
```

Submit your exercise 9 script by command line

```
oj submit ex9 <your_script_file>
```

- Input

```
I have a pen. I have an apple.
Uhh!! Apple-pen.

I have a pen. I have a pineapple.
Uhh!! Pineapple-pen.

Apple-pen. Pineapple-pen.
Uhh!! Pen pineapple apple pen.
```

- Output

```
a 3
an 1
apple 4
have 4
i 4
pen 8
pineapple 4
uhh 3
```

Assign9 Hint

[illegible]

Assign9 Hint

- [illegible]
  - [illegible]
    - [illegible]
    - [illegible]
  - [illegible]

    ```
    RESERVE NAME, FROM, TO, #tickets, SEAT(s)
    ```

  - [illegible]

    ```
    CANCEL NAME, FROM, TO, SEAT
    ```

  - [illegible]

    ```
    CANCEL NAME, SEAT
    ```

Assign9 Hint

- □□□□
  - □□

    ```
    RESERVE SUCCESSED!! -> NAME SEAT (FROM - TO)
    RESERVE FAILED.... (station information has something wrong)
    RESERVE FAILED.... (too many seats)
    RESERVE FAILED.... (repest seats)
    ```

  - □□

    ```
    CANCELLATION SUCCESSED!! SEAT (FROM - TO)
    CANCELLATION FAILED.... (cannot find the stations information)
    CANCELLATION FAILED.... (cannot find the seat information)
    ```

  - □□

    ```
    CHECK NAME SEAT -> (FROM - TO)
    CHECK FAILED.... (cannot find the reservation data)
    ```

Assign9 Hint

- [illegible]

  i. [illegible]4[illegible]

  ii. [illegible]

  iii. [illegible]

    - [illegible]
    - [illegible]

  iv. [illegible]

    - [illegible]4[illegible]

  v. [illegible]

    - [illegible]
    - [illegible]

  vi. [illegible]->[illegible]->[illegible]

Assign9 Hint

- [illegible]
  i. [illegible]"[illegible]"[illegible]"[illegible]"[illegible]"[illegible]"[illegible]
  ii. [illegible]"[illegible]"[illegible]
  iii. [illegible]
  iv. [illegible]

Assign9 Hint

- [illegible]
  i. [illegible]
  ii. [illegible]
  iii. [illegible]->[illegible]->[illegible]->[illegible]->[illegible]
  iv. [illegible]->[illegible]->[illegible]->[illegible]

Assign9 Hint

- □□(??)□□□
    1. □□□□□□□□□□□□□□□□□□□□□□□□
    2. □□`,`□□□□□□□□□□□□□□□□□□□□□□□
    3. □□□□□□□`gets`□□□□□□□□□□□
    4. □□□
    5. □□□□□□□□□□□□□OAO

Any Question?