

Computer Programming 1 Lab

2020-12-03



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 octal 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
- Return value:
 - On success, the total number of characters written is returned.
 - On failure, a negative number is returned.

Input/Output

```
#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.
- `other_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

```
#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;  
};
```

```
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

• • • • • • • • • • • •

- • • • • • • • • •
- • • • •
- • • • • • • • • • • • The the • • • • • •
- • • • • • • • •

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

██

Assign9 Hint

- 
 - 
 - 
 - 

-

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

-

CANCEL NAME, FROM, TO, SEAT

-

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)

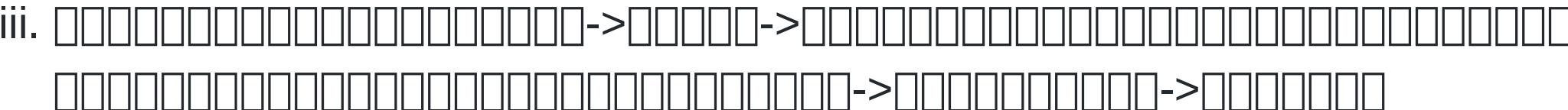
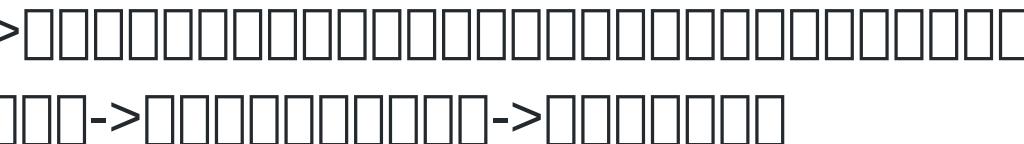
CHECK FAILED.... (cannot find the reservation data)

Assign9 Hint

- i. 4
- ii.
- iii.
 -
 -
- iv.
 - 40
- v.
 -
 -
- vi.

Assign9 Hint

Assign9 Hint

- 
 - 
 - 
 -  \rightarrow  \rightarrow 

 -  \rightarrow  \rightarrow 


Assign9 Hint

- $\lambda x.(\lambda y.y)(\lambda z.z)$
 - i. $\lambda x.(\lambda y.y)(\lambda z.z)(\lambda w.w)$
 - ii. $\lambda x.(\lambda y.y)(\lambda z.z)(\lambda w.w) \lambda$
 - iii. $\lambda x.(\lambda y.y)(\lambda z.z)(\lambda w.w) \lambda$ gets $\lambda x.(\lambda y.y)(\lambda z.z)(\lambda w.w) \lambda$
 - iv. $\lambda x.(\lambda y.y)(\lambda z.z)(\lambda w.w) \lambda$
 - v. $\lambda x.(\lambda y.y)(\lambda z.z)(\lambda w.w) \lambda$ OAO

Any Question?