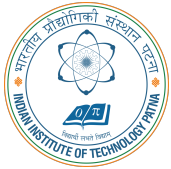
CS1101: Foundations of Programming

Additional topics



Dept. of Computer Science & Engineering

Indian Institute of Technology Patna

Topics covered

- **Input -** `scanf`
- **Output -** `printf`
- **Scope of variables**
- `enum` **specification**
- **Bitwise operation**
- **Multifile programming**

scanf - 1

- **Conversion character, we have seen `d, f, s, c`. Please consult books for more information**

`e,f,g`	floating point value
`h`	short int
`i`	decimal, hexadecimal, octal
`o`	octal
`u`	unsigned decimal integer
`X`	hexadecimal
`[...]`	data item is a string which may include white spaces

scanf - 2

- **The following prevents any previously entered extraneous characters from being assigned**

```
scanf(" %c %d %f", &c, &x, &y);
```

scanf - 3

- **Reading a string that can contain spaces and upper cases letters, eg. name of person**

```
char str[80];
scanf(" %[ ABCDEFGHIJKLMNOPQRSTUVWXYZ]s", str);
```

- **Input:** `IIT Patna` **— only** `IIT P` **will get stored in** `str`

scanf - 4

- **Reading a string that can contain spaces and upper cases letters, eg. name of person**

```
char str[80];
scanf(" %[^\n]s", str);
```

- **Input:** `IIT Patna` **— everything will get stored**
- **^ — denotes negation**

scanf - 5

```
int a, b, c;
scanf(" %3d %3d %3d", &a, &b, &c);
```

Case-1:

1 2 3

a=1, b=2, c=3

Case-2:

123 456 789

a=123, b=456, c=789

Case-3:

123456789

a=123, b=456, c=789

Case-4:

1234 5678 9

a=123, b=4, c=567

scanf - 6

```
float x;
char c;
scanf(" %5f %c", &x, &c);
printf("x=%f, c=%c", x, c);
```

Input:

```
256.875 T
```

Output:

```
x=256.8, c=7
```

scanf - 7

```
int mm, dd, yyyy;
scanf(" %d/%d/%d", &dd, &mm, &yyyy);
printf("Date: %d/%d/%d", dd, mm, yyyy);
```

Input:

10/11/2024

Output:

Date: 10/11/2024

```
int x; float y;
scanf(" %d A %f", &x, &y);
printf("x=%d y=%f", x, y);
```

Input:

8 A 3.4

Output:

x=8 y=3.4000000

scanf - 8

```
char c1, c2, c3;
scanf(" %c%c%c", &c1, &c2, &c3);
printf("c1=%c c2=%c c3=%c\n", c1, c2, c3);
```

Input:

```
a b c
```

Output:

```
c1=a c2=  c3=b
```

```
char c1, c2, c3;
scanf(" %c%1s%1s", &c1, &c2, &c3);
printf("c1=%c c2=%c c3=%c\n", c1, c2, c3);
```

Input:

```
a b c
```

Output:

```
c1=a c2=b c3=c
```

printf - 1

- **Conversion character, we have seen `d, f, s, c`. Please consult books for more information**

`e`	floating point with exponent
`f`	floating point without exponent
`g`	floating point with no trailing zeros
`i`	signed decimal integer
`o`	octal without leading zeros
`u`	unsigned decimal integer
`X`	hexadecimal without leading 0x
`%`	no argument, print %

printf - 2

```
float x=5000, y=0.0025;
printf("%f %f \n", x, y);
printf("%e %e \n", x, y);
```

Output:

```
5000.000000 0.002500
5.000000e+03 2.500000e-03
```

printf - 3

```
char str[80];
scanf("%[^\n]s",str);
printf("%s\n", str);
```

Input:

Happy Diwali!!

Output:

Happy Diwali!!

printf - 4

```
int i=12345; float x=345.687;
printf(":%3d:%8d:\n",i, i);
printf(":%3f:%8f:\n",x, x);
printf(":%7f:%7.3f:%7.1f:\n",x, x, x);
```

Output:

```
:12345:   12345:
:345.678000:345.678000:
:345.678000:345.678:  345.7:
```

printf - 5

- **Printing "Hello, world" (12 characters)**

```
:%s:          :hello, world:
:%10s:        :hello, world:
:%.10s:       :hello, wor:
:%-10s:       :hello, world:
:%.15s:       :hello, world:
:%-15s:       :hello, world   :
:%15.10s:     :     hello, wor:
:%-15.10s:    :hello, wor     :
```

Storage class

- `auto int a, b, c;` — **This is similar to local variables.**
- `register int a;` — **To store the value in (if possible) fast registers of the machine. Applicable to automatic variables**
- `static int a;` — **A static variable inside a function is similar to other local variables but unlike automatics, they remain in existence rather than coming and going each time the function is activated. Private but permanent storage**
- `extern int a;` — **Usually referred as global variable. Used for external linkage, more meaningful for multifile programming**

enum specification

- This is a kind of specification of constant integer
- **Example:**
  - `enum {NO, YES};` // `NO` is 0, and `YES` is 1

```
int x = 0;
if(x == NO){ ... }
```

  - `enum {Jan=1, Feb=2, ..., Dec=12};`

```
int x = 12;
if(x == Dec){ ... }
```

Bitwise operations

- In computer, all numbers are binary. C provides some support to perform bitwise manipulations
- & — bitwise AND, | — bitwise OR, ^ — bitwise XOR,
  << — left shift, >> — right shift, ~ — bitwise complement

```
unsigned int x; scanf("%u", &x); printf("%u = ",x);
unsigned int mask = 1 << 31; // left shift, multiplication by 2^31
for(int i=0; i<32; i++){
   putchar(x & mask ? '1' : '0' );
   x <<= 1; // shift left by 1
   if( i % 8 == 0) putchar(' ');
}
```

Multifile programming

myheader.h

```
#ifndef __MYHEADER_H__
#define __MYHEADER_H__
#include<stdio.h>
typedef struct {
   int x; int y;
} point;
void f1(int);
void f2(point);
#endif
```

main.c

```
#include "myheader.h"
extern int gVar;
int main(){
   point z = (point) {2, 3};
   f1(3);
   f2(z);
   printf("gVar=%d\n",gVar);
}
```

file1.c

```
#include "myheader.h"
int gVar=1;
void f1(int x){
   gVar++;
   printf("File: file1.c\n");
   printf("x=%d\n",x);
   printf("gVar=%d\n",gVar);
}
```

file2.c

```
#include "myheader.h"
extern int gVar;
void f2(point x){
   printf("gVar=%d\n",gVar);
   printf("x=%d y=%d\n",x.x, x.y);
   gVar++;
}
```

```
$> gcc main.c file1.c file2.c
```