

Fixed point vs floating point representation

Fixed point

- Position of radix point is fixed and is same for all numbers.
 - Lets say we have 3 digits after radix point.
 - $(0.120 \times 0.120)_{10} = (0.014)_{10}$
 - A digit is lost.
-

Floating point

- $1.20 \times (10)^{-1} \times 1.20 \times (10)^{-1} = 1.44 \times (10)^{-2}$
- Wider range of numbers can be represented.
- IEEE standard: 32 bits are split as follows:
 - First bit for sign.
 - Next 8 bits for exponent.
 - Next 23 bits for mantissa.
 - $(-39.9)_{10} = (-100111.11100)_2 = (-1.0011111100)_2 \times 2^5$.

Floats - different types

Type	Storage size	Value range
float	4 byte	1.2E-38 to 3.4E+38
double	8 byte	2.3E-308 to 1.7E+308
long double	10 byte	3.4E-4932 to 1.1E+4932

Output floats in C

```
printf(" %w.p f ", x);
```

- w.p is optional.
- w : total width of the field.
- p : precision (digits after decimal).

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

```
main() {
```

```
    float x = 2.00123;
```

```
    printf ("x = %5.4f\n", x);
```

```
    printf ("x = %8.7f\n", x);
```

```
}
```

Circumference of circle

```
#include<stdio.h>

main() {
    float radius;
    float circum;

    printf("Enter radius : ");
    scanf("%f", &radius);
    circum = 2* (22.0/7) * radius;

    printf ("radius = %f, circum = %f\n", radius, circum);
}
```

- How to print output only upto 2 decimals?

Circumference of circle – formatted output

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

```
main() {
```

```
    float radius;
```

```
    float circum;
```

```
    printf("Enter radius : ");
```

```
    scanf("%f", &radius);
```

```
    circum = 2* (22.0/7) * radius;
```

```
    printf ("radius = %5.2f, circum = %5.2f\n", radius, circum);
```

```
}
```

Output statement

```
printf (format-string, var1, var2, ..., varn)
```

Format string specifies

- How many variables to expect?
- Type of each variable.
- How many columns to use for printing? (width)
- What is the precision? (if applicable)
- **Common mistakes:**
 - comma missing after the double quotes.
 - mismatch in the actual number of variables given and those expected in the format string.

Formatted output

```
printf ( "%w.pC", x );
```

-
- **w**, **p** and **C** are place holders, can take different values.
 - **w**: width of the output. (optional)
 - **p**: precision of the output. (optional)
 - **C**: Conversion character.
 - d : integer
 - f : float
 - c : character
 - x : hexadecimal
 - o : octal
 - u : unsigned int
 - e : real decimal in exponent form

Input Statement

```
scanf (format-string, &var1, &var2, ..., &varn)
```

Format string specifies

- How many variables to expect?
- Type of each variable.
- **Common mistakes:**
 - comma missing after the double quotes.
 - mismatch in the actual number of variables given and those expected in the format string.
 - **& missing before the variable.**

Learnings so far..

- Variables in C.
- Data types in C - how they are stored. Why a programmer should be worried.
- Arithmetic operators. Precedence of operators.
- Assignment operator “=”
- Formatting the input and output - the printf and scanf
- What is coming up?
 - Compilation and Execution of C-programs.
 - More Programming.