

Integers in C and Storage

- We have used **int** and **float** data types till now.
- An integer variable is assigned 2 bytes (16 bits) to be stored. (Sometimes 4 bytes).
- In the 2s complement form this allows storage of values from

$$-2^{15} \text{ to } 2^{15} - 1$$

$$-32,768 \text{ to } 32,767$$

- There are limits to representation - we better choose the right type.
- What other data type can we use to store integers?
- **unsigned int, long, unsigned long.**

unsigned int

- Typically 4 bytes storage.
- Output an unsigned int: `printf("%u", x);`
- Input an unsigned int: `scanf("%u", &x);`
- Storage: binary format.

The Integers - The detailed Chart

int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

char

- Typically 1 byte storage.
- Every character has a unique code assigned to it (ASCII code).
A = 65, B = 66
- How do you find the code for a character?

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

```
main() {  
    printf ("%c\n", 65);  
}
```

- Output a character: `printf("%c", x);`
- Input a character: `scanf("%c", &x);`

float

- Typically 4 bytes storage.
- Output an unsigned int: `printf("%f ", x);`
- Input an unsigned int: `scanf("%f ", &x);`
- How are fractions stored?

Binary vs decimal fractions

- $(10.11)_2 = (1 \times 2) + (1 \times 1) + (1 \times \frac{1}{2}) + (1 \times \frac{1}{2^2}) = (2.75)_{10}$
- $(0.90625)_{10} = (\quad)_2$
- $(0.9)_{10} = (\quad)_2$

Decimal Fraction $\rightarrow$ Binary Fraction (1)

Convert $(0.90625)_{10}$ to binary fraction

0.90625			
$\times 2$			
1	0.8125	+ integer part	
$\times 2$			
1	0.625	+ integer part	
$\times 2$			
1	0.25	+ integer part	
$\times 2$			
1	0.5	+ integer part	
$\times 2$			
0	0	+ integer part 1	

$$\begin{aligned}0.90625 &= \frac{1}{2}(1+0.8125) \\&= \frac{1}{2}(1+ \frac{1}{2}(1+0.625)) \\&= \frac{1}{2}(1+ \frac{1}{2}(1+ \frac{1}{2}(1+0.25))) \\&= \frac{1}{2}(1+\frac{1}{2}(1+ \frac{1}{2}(1+\frac{1}{2}(0+0.5)))) \\&= \frac{1}{2}(1+\frac{1}{2}(1+\frac{1}{2}(1+\frac{1}{2}(0+\frac{1}{2}(1+0.0))))) \\&= \frac{1}{2}+1/2^2+1/2^3+0/2^4+1/2^5 \\&= (0.11101)_2\end{aligned}$$

Thus, $(0.90625)_{10} = (0.11101)_2$

Decimal Fraction $\rightarrow$ Binary Fraction (2)

Convert $(0.9)_{10}$ to binary fraction

$$\begin{array}{rcl} 0.9 & & \\ \times 2 & & \\ \hline 0.8 & + \text{integer part } 1 & \\ \times 2 & & \\ \hline 0.6 & + \text{integer part } 1 & \\ \times 2 & & \\ \hline 0.2 & + \text{integer part } 1 & \\ \times 2 & & \\ \hline 0.4 & + \text{integer part } 0 & \\ \times 2 & & \\ \hline 0.8 & + \text{integer part } 0 & \end{array}$$

For some fractions, we do not get 0.0 at any stage!

These fractions require an infinite number of bits!

Cannot be represented exactly!

Repetition

$$(0.9)_{10} = 0.1\textcolor{red}{1}1001100\textcolor{red}{1}1001100 \dots = 0.\overline{11100}$$

Binary vs decimal fractions

- $(10.11)_2 = (1 \times 2^1) + (0 \times 2^0) + (1 \times \frac{1}{2}) + (1 \times \frac{1}{2^2}) = (2.75)_{10}$
- $(0.90625)_{10} = (0.11101)_2$
- $(0.9)_{10} = (0.111001110011100\dots)_2$

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.
 - $(-39.9)_{10} = (-100111.11100)_2 = (-1.0011111100)_2 \times 2^5$.