# Size of integer

From Lecture 3 – Introduction to variables

2 bytes 4 bytes

Depends on the machine

2 bytes = 16 bits

4 bytes = 32 bits

More the size, more content it can hold.

# Wanna know size programmatically?

Use "sizeof" operator

```
#include <stdio.h>

int main()
{
    printf("%d", sizeof(int));
    return 0;
}
```

Note: sizeof is a unary operator and not a function.

Output:

```
4
```

Sizeof integer is 4 bytes in my machine. May be it is 2 bytes in your machine.

# Prerequisites

Decimal number system: Human Understandable number system.

Also called as base 10 number system.

Range: 0 to 9

| $10^2$ | $10^1$ | $10^0$ |
|---|---|---|
| 5 | 6 | 8 |

$$10^2 \times 5 + 10^1 \times 6 + 10^0 \times 8 = 500 + 60 + 8 = \boxed{568}$$

# Prerequisites

Binary number system: Machine Understandable number system.

Also called as base 2 number system.

Range: 0 to 1

$$\begin{array}{cccc} 2^3 & 2^2 & 2^1 & 2^0 \\ 1 & 1 & 0 & 0 \end{array}$$

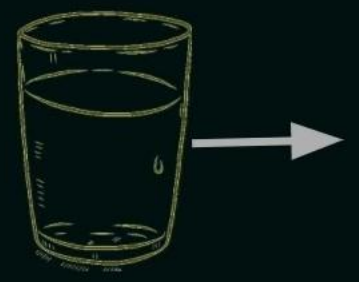
$$2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 0 = 8 + 4 + 0 + 0 = \boxed{12}$$

# Range of 4 bit data?

| | | |
|---|---|---|
| | $2^3$ $2^2$ $2^1$ $2^0$ | |
| 4 bit data: | 0 0 0 0 | Minimum Value = 0 |
| | 1 1 1 1 | Maximum Value = 15 |
| Range of 4 bit data: | 0 0 0 0 to 1 1 1 1 | |
| | 0 to 15 | |

$$\boxed{\text{Formula: } 2^n - 1}$$

# Range of integer

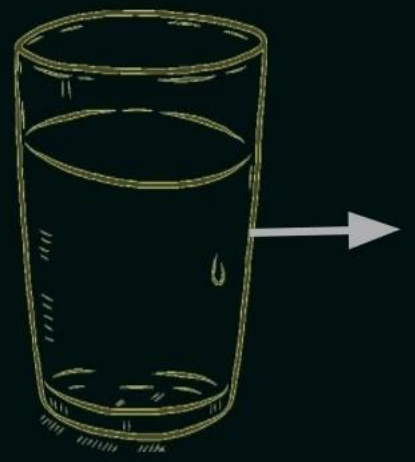


2 bytes
[16 bits]

Unsigned range: 0 to 65535 (by applying: $2^n - 1$)

Signed range: −32768 to +32767

2's complement range: $-(2^{n-1})$ to $+(2^{n-1} - 1)$



4 bytes
[32 bits]

Unsigned range: 0 to 4294967295 (by applying: $2^n - 1$)

Signed range: −2147483648 to +2147483647

# Long and short

If integer is 4 bytes, short int may be 2 bytes

On my computer:

```
#include <stdio.h>

int main()
{
    printf("%d", sizeof(short int));
    return 0;
}
```

Output:

2

# Long and short

If integer is 4 bytes, long int may be 8 bytes

On my computer:

```
#include <stdio.h>

int main()
{
    printf("%d", sizeof(long int));
    return 0;
}
```

Output:

```
8
```

# Long and short

sizeof (short) <= sizeof (int) <= sizeof (long)

Note: by default int some_variable_name; is signed integer variable.

Unsigned int some_variable_name; allows only positive values.

# Programming examples

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

int main()
{
    int var1 = INT_MIN;
    int var2 = INT_MAX;

    printf("range of signed integer is from: %d to %d", var1, var2);
    return 0;
}
```

Output:

```
range of signed integer is from: -2147483648 to 2147483647
```

# Programming examples

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

int main()
{
    unsigned int var1 = 0;
    unsigned int var2 = UINT_MAX;

    printf("range of unsigned integer is from: %u to %u", var1, var2);
    return 0;
}
```

## Output:

```
range of unsigned integer is from: 0 to 4294967295
```

# Programming examples

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

int main()
{
    short int var1 = SHRT_MIN;
    short int var2 = SHRT_MAX;

    printf("range of short signed integer is from: %d to %d", var1, var2);
    return 0;
}
```

Output:

```
range of short signed integer is from: -32768 to 32767
```

# Programming examples

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

int main()
{
    short unsigned int var1 = 0;
    short unsigned int var2 = USHRT_MAX;

    printf("range of short unsigned integer is from: %u to %u", var1, var2);
    return 0;
}
```

Output:

```
range of short unsigned integer is from: 0 to 65535
```

if sizeof (long int) = 4 bytes
then sizeof (long long int) = 8 bytes

else

if sizeof (long int) = 8 bytes
then sizeof (long long int) = 8 bytes

# Summary

1. sizeof (short) <= sizeof (int) <= sizeof (long).
2. Writing signed int some_variable_name; is equivalent to writing int some_variable_name;
3. %d is used to print "signed integer"
4. %u is used to print "unsigned integer"
5. %ld is used to print "long integer" equivalent to "signed long integer"
6. %lu is used to print "unsigned long integer"
7. %lld is used to print "long long integer"
8. %llu is used to print "unsigned long long integer"