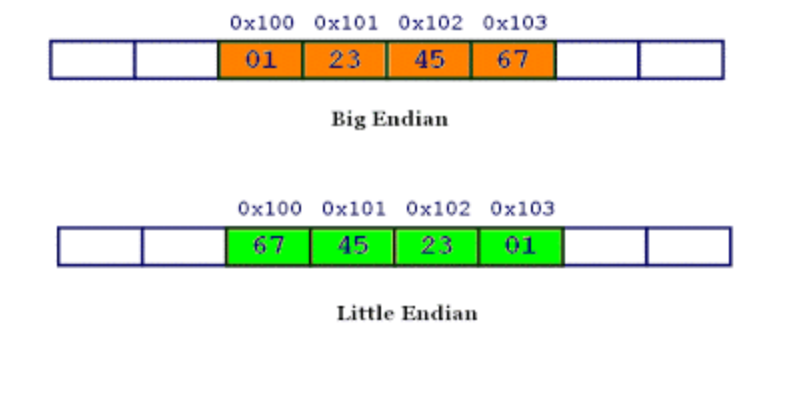
Fun with types

Endianness

In little endian machines, last byte of binary representation of the multi-byte data-type is stored first. On the other hand, in big endian machines, first byte of binary representation of the multi-byte data-type is stored first.

Suppose integer is stored as 4 bytes, then a variable x with value 0x01234567 will be stored as following.



Endianness

```
#include <stdio.h>
int main()
{
   unsigned int i = 1;
   char *c = (char*)&i;
   if (*c)
       printf("Little endian");
   else
       printf("Big endian");

   return 0;
}
```

Does it matter?

```
#include <stdio.h>
int main()
{
    unsigned char arr[2] = {0x01, 0x00};
    unsigned short int x = *(unsigned short int *) arr;
    printf("%d", x);
    return 0;
}
```

Puzzles

```
#include <stdio.h>
int main()
{
    double d = 3.1416;
    char ch = *(char *) &d;
    cout << ch << endl;
}
```

Puzzles

```
#include <stdio.h>
int main()
{
    short s = 45;
    double d = *(double *) &s;
    cout << d << endl;
}
```

Puzzles

```
#include <stdio.h>
struct fraction {
    int num;
    int denom;
};

fraction pi;

int main()
{
    pi.num = 22;
    pi.denom = 7;
    cout << pi.denom << endl;
    ((fraction *) &(pi.denom))->num = 12;
    cout << pi.denom <<endl;
}
```

Puzzles

```
struct student {
    char *name;
    char suid[8];
    int numUnits;
};

student pupils[4];

int main () {
    pupils[0].numUnits = 21;
    strcpy (pupils[0].suid, "ABCDEFG");
    strcpy (pupils[1].suid, "40415xx");
    pupils[2].name = strdup ("Adam");;
    pupils[3].name = pupils[0].suid + 6;
    strcpy (pupils[3].name, "123456");
}
```