Data types

Data Representation in Computer Memory

Data representation in computer memory is a fundamental concept in computer science. It involves how information is stored, organized, and accessed within a computer's memory system. This process is crucial for the execution of programs and the manipulation of data.

At the heart of data representation lies the binary system. Unlike humans, who primarily use the decimal system (base 10) with digits ranging from 0 to 9, computers operate on the binary system (base 2), which uses only two digits - 0 and 1. This is due to the electronic nature of computer circuits, which can easily distinguish between two states - on and off.

In this binary system, the smallest unit of data is called a "bit," which can hold either a 0 or a 1. Eight bits together form a "byte," which is the basic unit for data storage and processing. A byte can represent up to 256 different values (2^8).

Data comes in various forms, each with its own representation. For instance, integers, which are whole numbers, can be represented using either a signed or unsigned format. A signed integer includes both positive and negative numbers and employs a sign bit to indicate the sign. In contrast, an unsigned integer only represents positive numbers and doesn't use a sign bit.

For real numbers, the floating-point representation is used. It divides the data into a mantissa (which holds the significant digits) and an exponent (which determines the scale of the number). This allows the representation of a wide range of values, from very small to very large, with a certain level of precision.

Characters, on the other hand, are represented using character encoding standards like ASCII (American Standard Code for Information Interchange) or Unicode. Each character is assigned a unique numeric value, allowing computers to store and process text.

Memory in a computer is organized into bytes, each with a unique address. This enables the computer to locate and access specific data when needed. Moreover, data is stored in either big-endian or little-endian format, which determines the order in which the bytes of a multi-byte value are stored in memory.

Programs dynamically allocate memory during runtime to store variables and data structures. This allows for flexibility in managing memory resources based on the program's requirements.

Type	Size (byte)	Format
int	at least 2, generally 4	%d, %i
char	1	%c
float	4	%f
double	8	%lf
short int	2 generally	%hd

Type	Size (byte)	Format
unsigned int	at least 2, generally 4	%u
long int	at least 4, generally 8	%ld, %li
long long int	at least 8	%lld, %lli
unsigned long int	at least 4	%lu
unsigned long long int	at least 8	%llu
signed char	1	%c
unsigned char	1	%c
long double	Au moins 10, generally 12 or 16	%Lf

Type conversion

Implicit Type Conversion (Automatic): Performed by the compiler itself, Generally occurs when multiple data types are present in an expression. A type conversion (type promotion) takes place to prevent data loss. All variable data types are upgraded to the data type of the variable with the largest data type.

Explicit Type Conversion (Automatic): This process is also known as type casting and is defined by the user. Here, the user can explicitly specify the result to be transformed into a particular data type.

```c

        // Link
        #include "stdio.h"

        // Main() Function
        int main(void)
        {
        //Implicit Type Conversion
        puts("Implicit Type Conversion");
        int x=-3;
        printf("x: %i\n", x);
        printf("x: %u\n", x);

        char c='b';
        printf("x: %c\n", c);
        printf("x: %i\n", c);
        printf("x: %c\n", c+x);
        printf("x: %i\n", c+x);

        //Explicit Type Conversion
        puts("Explicit Type Conversion");
        float x2=1.2;
        float sum = x2+1;
        float sum2 = (int)x2 +1;
        printf("sum: %f\n", sum);
        printf("sum2: %f\n", sum2);
        return 0;
        }
```