# Data and Its Type

## Name and Location

- Data is stored in the memory.
- In a machine instruction, a memory location is identified by its address.
- In a high-level language[a] a location is identified with a name, called a variable. A variable is bound to a memory location.

[a] Imperative programming language e.g. Fortran, Algol, Pascal, C, C++ etc.

# Name and Location

- Data can be read from a memory location and a memory location can also be updated.
- This facility is made available in a high-level language by using a variable as an expression and making an assignment to a variable.

# Types of Data

- There can be data of many different types e.g. whole numbers, integers, rational numbers, real numbers, complex numbers, vectors, matrices, characters etc.
- In the machine hardware everything is encoded as strings of binary digits (0 and 1) of finite lengths.

## Types of Data

In the machine a few primitive types of data are differentiated and processed by different instructions and pieces of hardware e.g. an integer data is processed in the ALU and a fractional data is processes in the FPU.

## Types of Data

- This gets reflected in the built-in or primitive data types of a high level language.
- Modern high level languages also provide facility to construct complex data types using type constructors.

# A Few Built-in Data Types in C

| | | |
|---|---|---|
| int | float | char |
| unsigned int | | unsigned char |
| long long int | double | |

## Simple Variable Declaration in C

```
char upperCase, grade = 'B';
```

```
int count, index = 9;
```

```
float cgpa = 9.5, interest;
```

- `char`, `int`, `float` are a few built-in data types of C language.
- `upperCase` and `grade` are variables[a] of type `char`.
- `grade = 'B'` initializes the variable `grade` to the character code of `'B'`.

[a] A variable names or any C identifier follows a convention; letter, underscore followed by letter, underscore or digit.

- `count` and `index` are variables of type `int`.
  `index = 9` initializes the variable to the binary representation of 9.
- `cgpa` and `interest` are variables of type `float`.
  `cgpa = 9.5` initializes the variable to the binary representation of 9.5 (different from int).

## int is not Integer

- An integer data may be arbitrarily large, but the C language data type `int` has only 32 binary digits or bit positions, for its value.
- The range of `int` data is $-2^{31} = -2147483648$ to $2^{31} - 1 = 2147483647$.
- The representation is in 2's complement form.

## float is an Approximation of Real

- A real numbers may have infinite information content (irrational numbers) that cannot be stored in a finite computer.
- Data type `float` is an approximation of real numbers. It also has a fixed 32-bit size, but the representation is different from `int` (IEEE 754 single precession)[a].

[a] The representations of 10 and 10.0 are different inside a computer.

## Range of `float`

- The smallest and the largest magnitudes of `float` data are approximately $1.401298 \times 10^{-45}$ and $3.402823 \times 10^{38}$ respectively.
- Special `float` values such as nan (not a number e.g. $\sqrt{-1}$) and inf (infinity - 1.0/0.0) are defined to handle error in floating point operation.

## char is a Short Integer

- In the binary world of computer every data, primitive or constructed, is encoded as a bit string of finite length.
- The useful set of characters are encoded as 8-bit (one byte) or 16-bit integers.
- The C language uses 8-bit ASCII[a] encoding.

[a] ASCII stand for American Standard Code for Information Interchange.

# A few ASCII Codes

| char | decimal | binary | hex |
|---|---|---|---|
| 0 | 48 | 0011 0000 | 30 |
| 9 | 57 | 0011 1001 | 39 |
| A | 65 | 0100 0001 | 41 |
| Z | 90 | 0101 1010 | 5a |
| a | 97 | 0110 0001 | 61 |
| z | 122 | 0111 1010 | 7a |

## Binary to Hex

It is tedious to write a long string of binary digits. A better way is to use radix-16 or hexadecimal (Hex) number system with 16 digits $\{0, 1, \cdots, 9, A(10), B(11), C(12), D(13), E(14), F(15)\}$.

## Binary to Hex

To convert from binary to hex representation, the bit string is grouped in blocks of 4-bits (nibble) from the least significant side. Each block is replaced by the corresponding hex digit.

## Binary to Hex

$$0011 \quad 1110 \quad 0101 \quad 1011 \quad 0001 \quad 1101 \quad 0110 \quad 1001$$

$$\Downarrow$$

$$3 \quad E \quad 5 \quad B \quad 1 \quad D \quad 6 \quad 9$$

We write 0x3E5B1D69 (lower case letters can also be used) for a hex constant in C language.

## `int` Data: an Example

- $7529_D \equiv$ 0000 0000 0000 0000 0001 1101 0110 $1001_B$ $\equiv$ 0x00001D69 = 0x1D69
- $-7529_D \equiv$ 1111 1111 1111 1111 1110 0010 1001 $0111_B$ $\equiv$ 0xFFFFE297

We shall discuss about this representation afterward.

## `float` Data: an Example

- $7529.0_D \Rightarrow$ 0 1000 1011 110 1011 0100 1000 0000 $0000_B$
- $-7529.0_D \Rightarrow$ 1 1000 1011 110 1011 0100 1000 0000 $0000_B$

This representations are different from that of 7529 or −7529.

## char Data: an Example

- 'A' $\equiv$ 0100 $0001_B$ $\equiv$ 0x41
- '1' $\equiv$ 0011 $0001_B$ $\equiv$ 0x21.

It is not same as 1 or 1.0.

## A Few Other Built-in Types of C

- unsigned int (unsigned) - 32-bit unsigned binary, 0 to $2^{32} - 1 = 4294967295$.
- long int is same as int.
- long long int - 64-bit signed binary, $-2^{63} = 9223372036854775808$ to $2^{63} - 1 = 9223372036854775807$.

## A Few Other Built-in Types of C

- double - 64-bit IEEE 754 double precession format.

## Constants of Primitive Types

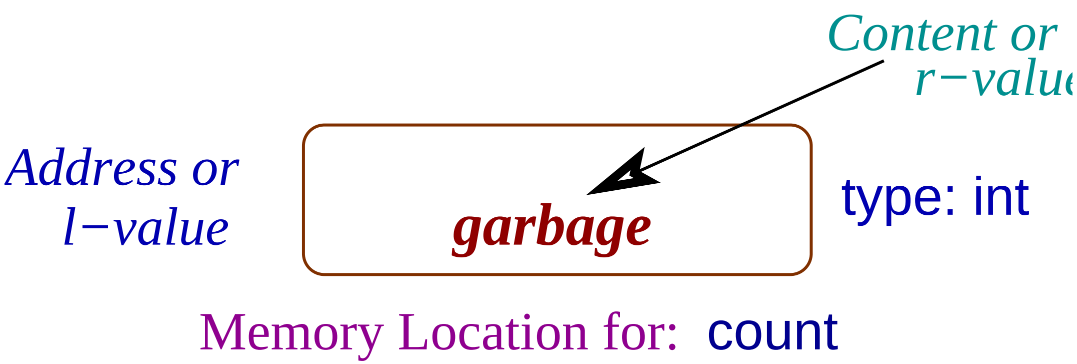
- int: 123, −123
- float: 1.23, −1.23e−02
- char: 'A', '5', '%'

A floating-point constant is often taken in double precision format.

## A Variable and Its Memory Location

```
int count;
```

int count;


Content or
r−value
Address or
l−value
garbage
type: int
Memory Location for: count

## Note

- Either the compiler generates code to allocate memory or it is allocated when the process image (a.out for example) is loaded.
- The allocated memory (location) has an address or l-value.
- The allocated space is of fixed size to store the data of the specified type. It is 4-bytes for `int`.

## Note

- Unless initialized, the content or the *r-value* is undefined after the declaration.
- The content or the *r-value* can be initialised and updated.

```
int count = 10;
count = 100;
count = 2*count + 5;
```

## Note

- The address or the l-value of a variable can be extracted using the unary operator '&' (`&count`).
- This value of a location can be stored in another variable of type `int *` known as pointer type.

```
int count = 10, *cP;
cP = &count;
```

## Memory Locations for Other Types

```
float cgpa;
char grade;
```

- Memory allocations are similar for other data types e.g. `float` and `char`.
- The only difference is the size (type) of the location.

## **Constant:** `const`

A declaration can be qualified to define a name of a constant.

```
const double pi = 3.14159265358979323846
```

In this case we cannot modify `pi`, its value is stored in the read-only memory segment.

## Constant: `const`

```
#include <stdio.h>
int main()
{
    const double pi = 3.141592653589793 2;

    pi = pi + 1;
    return 0;
} // eight.c
```

## Constant: `const`

```
$ cc -Wall eight.c
eight.c:  In function ‘main’:
eight.c:9:  error:  assignment of
read-only variable ‘pi’
```

## Reading char Data

```
#include <stdio.h>
int main() {
    char c, d;

    printf("Enter two characters: ");
    scanf("%c", &c);
    scanf("%c", &d);
    printf("%c..%c\n", c, d);
    return 0;
} // charRead.c
```

This program is expected to read two characters from two lines.

## Reading char Data

```
$ cc -Wall charRead.c
$ a.out
Enter two characters: 1
1..

$
```

It does not read the second character. The reason is that pressing of 'Enter' key injects a non-printable character '\n' in the input stream.

## Invisible to Visible

Replace: `printf("%c..%c\n", c, d);`
by: `printf("%c..%d\n", c, d);`

```
$ cc -Wall charRead.c
$ a.out
Enter two characters: 1
1..10

$
```

## Invisible to Visible

To read proper input,
Replace: `scanf("%c", &d);`
by: `scanf(" %c", &d);` Note the gap.

```
$ cc -Wall charRead.c
$ a.out
Enter two characters: 1
2
1..2
```

The 'gap' is matched with '`\n`'.