Variables

Variables are memory locations(storage area) in the C programming language.

The primary purpose of variables is to store data in memory for later use. Unlike constants which do not change during the program execution, variable value may change during execution. If you declare a variable in C, that means you are asking the operating system to reserve a piece of memory with that variable name.

Variable Declaration

Syntax:

type variable_name;

or

type variable_name, variable_name, variable_name;

Variable Declaration and Initialization

Example:

```
int    width, height=5;
char   letter='A';
float  age, area;
double d;

/* actual initialization */width = 10;
age = 26.5;
```

Variable Assignment

A variable assignment is a process of assigning a value to a variable.

Example:

```
int width = 60;
int age = 31;
```

There are some rules on choosing variable names

1. A variable name can consist of Capital letters A-Z, lowercase letters a-z, digits 0-9, and the underscore character.
2. The first character must be a letter or underscore.
3. Blank spaces cannot be used in variable names.
4. Special characters like #, $ are not allowed.

5. C keywords cannot be used as variable names.
6. Variable names are case sensitive.
7. Values of the variables can be numeric or alphabetic.
8. Variable type can be char, int, float, double, or void.

Example:

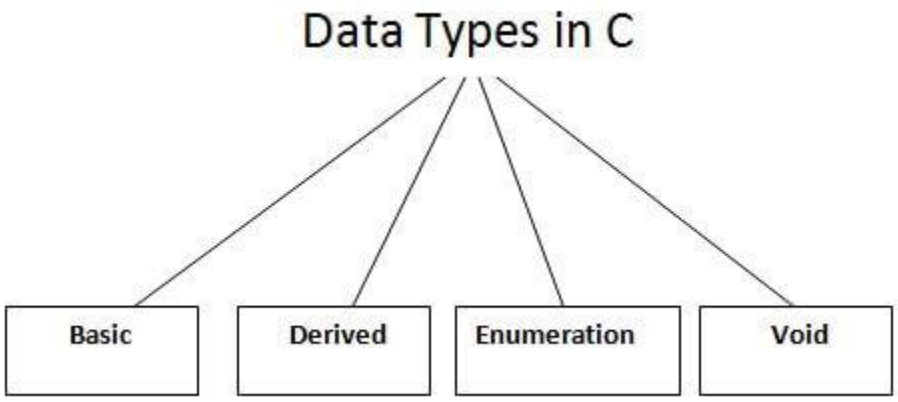
```
#include<stdio.h>

void main()
{
   int age = 33; // c program to print value of a variable
   printf("I am %d years old.\n", age);
}
```

Program Output:

```
I am 33 years old.
```

Data Types

A data type specifies the type of data that a variable can store such as integer, floating, character, etc.



There are the following data types in C language.

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	void

Basic Data Types

The basic data types are integer-based and floating-point based. C language supports both signed and unsigned literals.

The memory size of the basic data types may change according to 32 or 64-bit operating system.

Let's see the basic data types. Its size is given according to 32-bit architecture.

Data Types	Memory Size	Range
char	1 byte	−128 to 127
signed char	1 byte	−128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	−32,768 to 32,767
signed short	2 byte	−32,768 to 32,767
unsigned short	2 byte	0 to 65,535
int	2 byte	−32,768 to 32,767
signed int	2 byte	−32,768 to 32,767
unsigned int	2 byte	0 to 65,535
short int	2 byte	−32,768 to 32,767
signed short int	2 byte	−32,768 to 32,767
unsigned short int	2 byte	0 to 65,535
long int	4 byte	-2,147,483,648 to 2,147,483,647
signed long int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
float	4 byte	
double	8 byte	
long double	10 byte	

1. Character - ASCII character set or generally a single alphabet like 'a', 'B', etc.
2. Integer - Used to store whole numbers like 1, 2, 100, 1000, etc.
3. Floating-point - Decimal point or real numbers values like 99.9, 10.5, etc.
4. Double - Very large numeric values which are not allowed in Integer or Floating point type.
5. Void - This means no value. This data type is mostly used when we define functions.

There are different keywords to specify these data types, the keywords are:

Datatype	Keyword
Character	char
Integer	int
Floating-point	float
Double	double
Void	void

Each data type has a size defined in bits/bytes and has a range for the values that these data types can hold.

Size of different Data types

The size for different data types depends on the compiler and processor types, in short, it depends on the Computer on which you are running the C language and the version of the C compiler that you have installed.

char is 1 byte

The char datatype is 1 byte in size or 8 bits. This is mostly the same and is not affected by the processor or the compiler used.

int can be 2 bytes/4 bytes

There is a very easy way to remember the size for int datatype. The size of int datatype is usually equal to the word length of the execution environment of the program. In simpler words, for a 16-bit environment, int is 16 bits or 2 bytes, and for a 32-bit environment, int is 32 bits or 4 bytes.

float is 4 bytes

The float datatype is 4 bytes or 32 bits in size. It is a single-precision data type that is used to hold decimal values. It is used for storing large values.
float is a faster data type as compared to double, because double data type works with very large values, hence it is slow.

double is 8 bytes

The double datatype is 8 bytes or 64 bits in size. It can store values that are double the size of what a float data type can store, hence it is called double.
In the 64 bits, 1 bit is for sign representation, 11 bits for the exponent, and the rest 52 bits are used for the mantissa.
The double data type can hold approximately 15 to 17 digits, before the decimal and after the decimal.

void is 0 bytes

The void data type means nothing, hence it doesn't have a size.
Before moving on to the range of values for these data types, there is one more important concept to learn, which is Datatype modifiers.

Example:

```
// C program to read strings

#include<stdio.h>
```

```
int main()
{
	// declaring string
	char str[50];

	// reading string
	scanf("%s",str);

	// print string
	printf("%s",str);

	return 0;
}
```

Datatype Modifiers

In the C language, there are 4 data type modifiers that are used along with the basic data types to categorize them further.

For example, if you say, there is a playground, the other person will know that there is a playground, but you can be more specific and say, there is a Cricket playground or a Football playground, which makes it even more clear for the other person.
Similarly, there are modifiers in the C language, to make the primary data types more specific.
Following are the modifiers:

1. signed
2. unsigned
3. long
4. short

As the name suggests, signed and unsigned are used to represent the signed(+ and -) and unsigned(only +) values for any data type. And long and short affects the range of the values for any datatype.
For example, signed int, unsigned int, short int, long int, etc. are all valid data types in the C language.

Type	**Format Specifier**
char	%c
unsigned char	%c
signed char	%c
int	%d, %i
unsigned int	%u
signed int	%d, %i
short int	%hd
unsigned short int	%hu
signed short int	%hd
long int	%ld, %li

long long int	%lld, %lli
signed long int	%ld, %li
unsigned long int	%lu
unsigned long long int	%llu
float	%f
double	%lf
long double	%Lf

As you can see in the table above, with different combinations of the data type and modifiers the range of value changes.
When we want to print the value for any variable with any data type, we have to use a format specifier in the printf() statement.

What happens if the value is out of Range?

Well, if you try to assign a value to any datatype which is more than the allowed range of value, then the C language compiler will give an error. Here is a simple code example to show this,

```
#include <stdio.h>

int main() {
  // allowed value up to 65535
  unsigned short int x = 65536;

  return 0;
```

```
}
```

```
warning: large integer implicitly truncated to unsigned type [-Woverflow]
unsigned short int x = 65536;
^
```

When a type modifier is used without any data type, then the int data type is set as the default data type. So, unsigned means unsigned int, signed means signed int, long means long int, and short means short int.

What does signed and unsigned means?

In simple words, the unsigned modifier means all positive values, while the signed modifier means both positive and negative values.
When the compiler gets a numeric value, it converts that value into a binary number, which means a combination of 0 and 1. For example, 32767 in binary is 01111111 11111111, and 1 in binary is 01 (or 0001), 2 is 0010, and so on.
In the case of a signed integer, the highest order bit or the first digit from left (in binary) is used as the sign flag. If the sign flag is 0, the number is positive, and if it is 1, the number is negative.
And because one bit is used for showing if the number is positive or negative, hence there is one less bit to represent the number itself, hence the range is less.
For signed int, 11111111 11111111 means -32,767 and because the first bit is a sign flag to mark it as a negative number, and the rest represent the number. Whereas in the case of unsigned int, 11111111 11111111 means 65,535.

Example:

```
#include <stdio.h>
int main()
{
   int a = 4000; // positive integer data type
   float b = 5.2324; // float data type
   char c = 'Z'; // char data type
   long d = 41657; // long positive integer data type
   long e = -21556; // long -ve integer data type
```

```
    int f = -185; // -ve integer data type
    short g = 130; // short +ve integer data type
    short h = -130; // short -ve integer data type
    double i = 4.1234567890; // double float data type
    float j = -3.55; // float data type
}
```

The storage representation and machine instructions differ from machine to machine. sizeof operator can be used to get the exact size of a type or a variable on a particular platform.

Derived Data Types

Data Types	**Description**
Arrays	Arrays are sequences of data items having homogeneous values. They have adjacent memory locations to store values.
References	Function pointers allow referencing functions with a particular signature.
Pointers	These are powerful C features which are used to access the memory and deal with their addresses.

User Defined Data Types

C allows the feature called type definition which allows programmers to define their identifier that would represent an existing data type. There are three such types:

Data Types	**Description**
Structure	It is a package of variables of different types under a single name. This is done to handle data efficiently. "struct" keyword is used to define a structure.
Union	These allow storing various data types in the same memory location. Programmers can define a union with different members, but only a single member can contain a value at a given time. It is used for
Enum	Enumeration is a special data type that consists of integral constants, and each of them is assigned with a specific name. "enum" keyword is used to define the enumerated data type.