# C Programming

Kasun De Zoysa

# Keywords

Keywords are predefined, reserved words used in programming that have special meanings to the compiler. Keywords are part of the syntax and they cannot be used as an identifier.
For example:

```
int money;
```

Here, int is a keyword that indicates '**money**' is a **variable** of type **integer**.

# Keywords in C Language

As C is a case sensitive language, all keywords must be written in **lowercase**.

| Keywords in C Programming | | | |
|---|---|---|---|
| auto | break | case | char |
| const | continue | default | do |
| double | else | enum | extern |
| float | for | goto | if |
| int | long | register | return |
| short | signed | sizeof | static |
| struct | switch | typedef | union |
| unsigned | void | volatile | while |

# C Identifiers

Identifier refers to name given to entities such as **variables**, **functions**, **structures** etc.

Identifier must be **unique**. They are created to give unique name to a entity to identify it during the execution of the program.
For example:

```
int money;
double accountBalance;
```

Here, **money** and **accountBalance** are identifiers.

Identifier names must be different from keywords. You cannot use **int** as an identifier because int is a keyword.

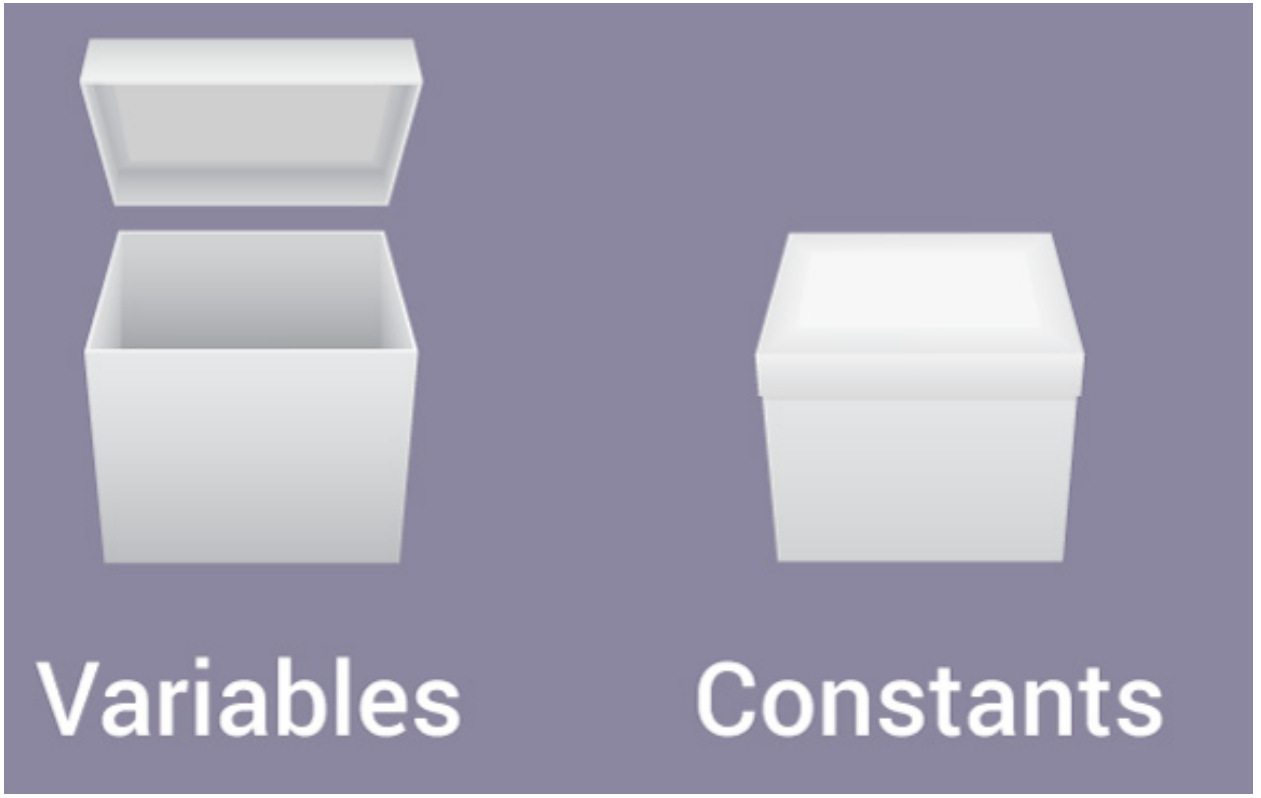
# Rules for writing an identifier

1)A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.

2)The first letter of an identifier should be either a letter or an underscore. However, it is discouraged to start an identifier name with an underscore.

3)There is no rule on length of an identifier.However, only the first 31 characters of a identifier are checked by the compiler.

# Variables and Constants



# Variables

In programming, a variable is a container (storage area) to hold data. To indicate the storage area, each variable should be given a unique name (identifier). Variable names are just the symbolic representation of a memory location.
For example:

```
int playerScore = 95;
```

Here, **playerScore** is a variable of integer type.
The variable is assigned value: 95.

The value of a variable can be changed, hence the name 'variable'.

# Rules for naming a variable in C

1)A variable name can have letters (both uppercase and lowercase letters), digits and underscore only.

2)The first letter of a variable should be either a letter or an underscore. However, it is discouraged to start variable name with an underscore. It is because variable name that starts with an underscore can conflict with system name and may cause error.

3)There is no rule on how long a variable can be. However, only the first 31 characters of a variable are checked by the compiler. So, the first 31 letters of two variables in a program should be different.

**C is a strongly typed language. What this means it that, the type of a variable cannot be changed.**

# Constants/Literals

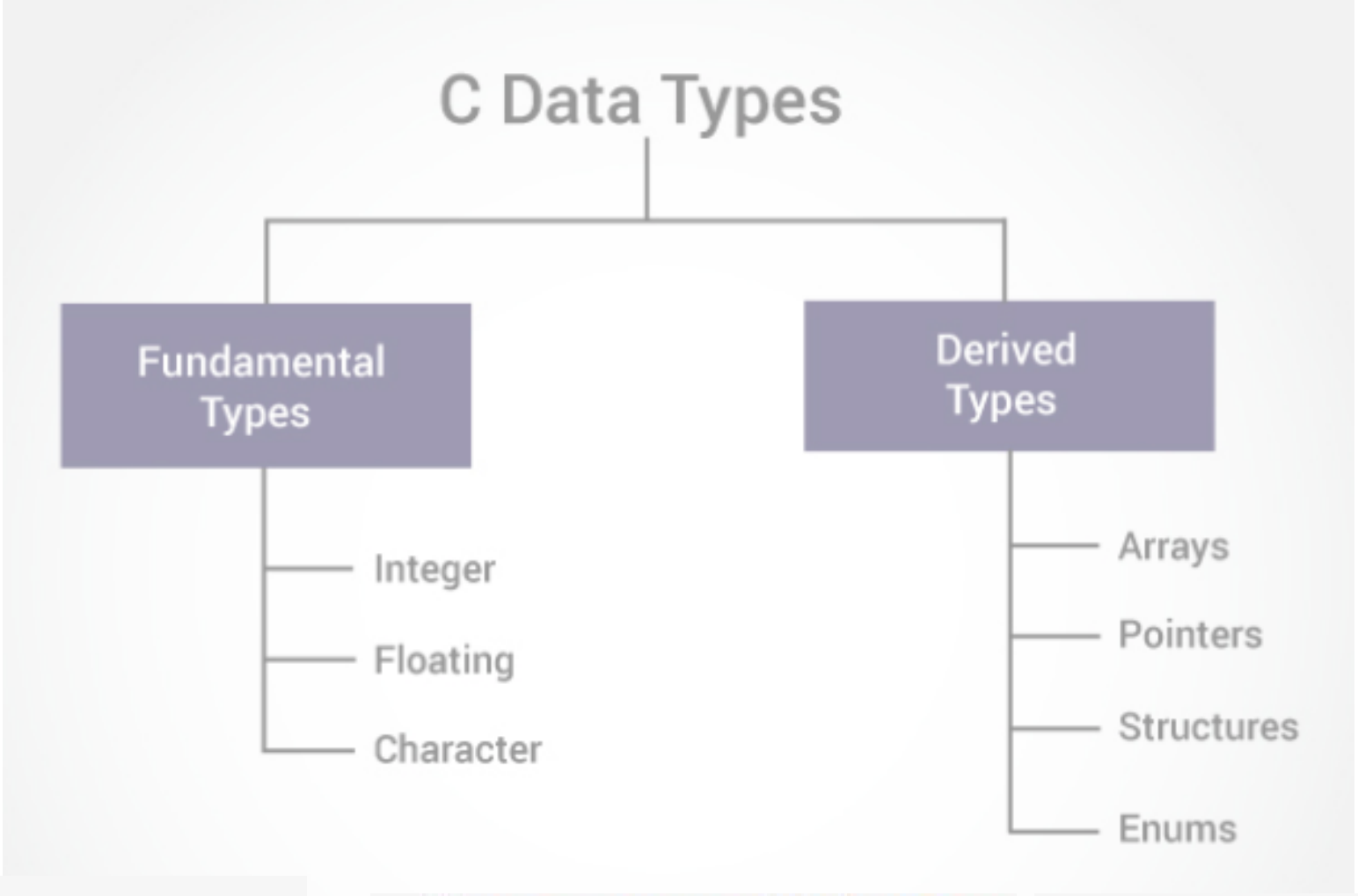
A constant is a value or an identifier whose value cannot be altered in a program. For example: 1, 2.5, "C programming is easy", etc.

As mentioned, an identifier also can be defined as a constant.

```
const double PI = 3.14
```

Here, PI is a constant. Basically what it means is that, PI and 3.14 is same for this program.

# Data Types




```
int id, age;
```

```
float accountBalance;
double bookPrice;
```

```
char test = 'h';
```

# int - Integer data types

Integers are whole numbers that can have both positive and negative values but no decimal values.

**Example**: 0, -5, 10

In C programming, keyword **int** is used for declaring integer variable.

**For example**:

```
int id;
```

Here, id is a variable of type integer.

You can declare multiple variable at once in C programming.

**For example:**

```
int id, age;
```

# Integer data types

The size of int is either 2 bytes (In older PC's) or 4 bytes.

If you consider an integer having size of 4 byte( equal to 32 bits), it can take $2^{32}$ distinct states as: $-2^{31}$,$-2^{31}+1$, ...,-2, -1, 0, 1, 2, ..., $2^{31}-2$, $2^{31}-1$.

If you try to store larger number than $2^{31}-1$,

i.e,+2147483647 and smaller number than $-2^{31}$,

i.e, -2147483648, program will not run correctly.

Similarly, int of 2 bytes, it can take $2^{16}$ distinct states from $-2^{15}$ to $2^{15}-1$.

# float - Floating types

Floating type variables can hold real numbers such as: 2.34, -9.382, 5.0 etc.

You can declare a floating point variable in C by using either **float** or **double** keyword.

**For example:**

```
float accountBalance;
double bookPrice;
```

Here, both accountBalance and bookPrice are floating type variables.

In C, floating values can be represented in exponential form as well.

For example:

```
float normalizationFactor = 22.442e2;
```

# Difference - float and double

The size of float (single precision float data type) is 4 bytes.

And the size of double (double precision float data type) is 8 bytes.

Floating point variables has a precision of **6 digits** whereas the precision of double is **14 digits**.

# char - Character types

Keyword **char** is used for declaring character type variables.

For example:

```
char test = 'h';
```

Here, test is a character variable.
The value of test is 'h'.

The size of character variable is 1 byte.

# Size qualifiers

Size qualifiers alters the size of a basic type.

There are two size qualifiers, **long** and **short**.

For example:

```
long double i;
```

The size of double is 8 bytes.

However, when long keyword is used, that variable becomes 10 bytes.

# Sign qualifiers

Integers and floating point variables can hold both negative and positive values.

However, if a variable needs to hold positive value only, **unsigned** data types are used.

**For example:**

```
// unsigned variables cannot hold negative value
unsigned int positiveInteger;
```

It is not necessary to define variable **signed** since a variable is signed by default.

An integer variable of 4 bytes can hold data from $-2^{31}$ to $2^{31}-1$. However, if the variable is defined as **unsigned**, it can hold data from 0 to $2^{32}-1$.
**It is important to note that, sign qualifiers can be applied to int and char types only.**

# Constant qualifiers

An identifier can be declared as a constant.
To do so **const** keyword is used.

```
const int cost = 20;
```

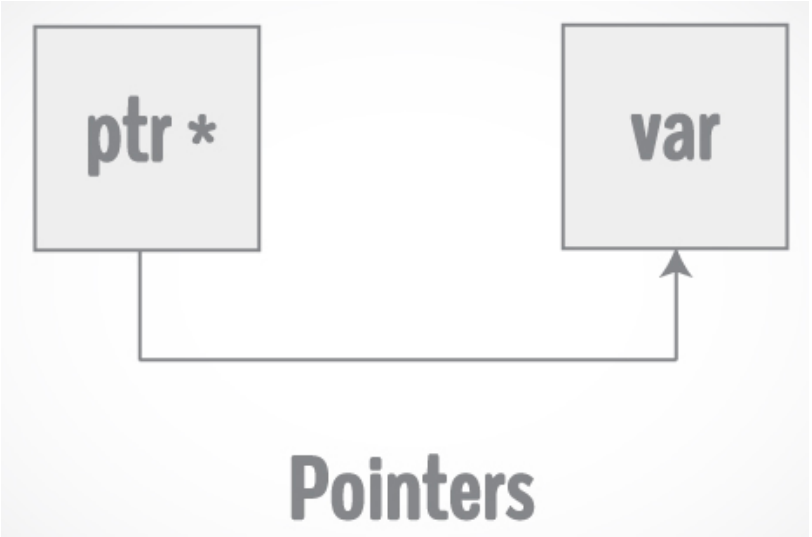
**The value of cost cannot be changed in the program.**

# Volatile qualifiers

A variable should be declared **volatile** whenever its value can be changed by some external sources outside the program.

Keyword **volatile** is used for creating volatile variables.

# Variable and Pointers




Pointers are powerful features of C and (C++) programming that differentiates it from other popular programming languages like: Java and Python.

Pointers are used in C program to access the memory and manipulate the address.

# Address in C

Before you get into the concept of pointers, let's first get familiar with address in C.

If you have a variable var in your program, &var will give you its address in the memory, where & is commonly called the reference operator.

You must have seen this notation while using scanf() function. It was used in the function to store the user inputted value in the address of var

```
scanf("%d", &var);
```

# Address in C

```c
/* Example to demonstrate use of reference operator in C programming. */
#include <stdio.h>
int main()
{
   int var = 5;
   printf("Value: %d\n", var);
   printf("Address: %u", &var);   //Notice, the ampersand(&) before var.
   return 0;
}
```

**Output**

```
Value: 5
Address: 2686778
```

**Note:** You may obtain different value of address while using this code.

In above source code, value 5 is stored in the memory location 2686778. var is just the name given to that location.

# Pointer Variables

In C, there is a special variable that stores just the address of another variable. It is called **Pointer variable** or, simply, a **pointer**.

Declaration of Pointer

```
data_type* pointer_variable_name;
int* p;
```

Above statement defines, p as pointer variable of type int.

# Reference operator (&) and Dereference operator (*)

As discussed, **&** is called reference operator. It gives you the address of a variable.

Likewise, there is another operator that gets you the value from the address, it is called a dereference operator (*****).

**Note:** The * sign when declaring a pointer is not a dereference operator. It is just a similar notation that creates a pointer.

# Problems

1. Write a C program to multiply two floating point numbers

2. Write a C program that computes the area of a disk.

3. Write a C Program to swap two numbers