Variable

Variable: A **variable** is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times. It is a way to represent memory location through symbol so that it can be easily identified. The variable length is max 31 character.

Rules for defining variables

- A variable can have alphabets, digits, and underscore.
- A variable name can start with the alphabet, and underscore only. It can't start with a digit.
- No whitespace is allowed within the variable name.
- A variable name must not be any reserved word or keyword, e.g. int, float, etc.
- A variable must not contain any special character(e.g. !@$*.'[] etc) except underscore(_).

Examples of some valid variable:
num, Num, _num, _Num, num1, Num1, _num1, _Num1, _1num, _1Num, _num_, number_to_add etc.

Examples of some invalid variable:
1num, number to add, 1_num, num-to-add, num@ etc.

Types of Variable

Types of Variables in C

- There are many types of variables in c:
- local variable
- global variable
- static variable
- automatic variable
- external variable

Data Type

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

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

Example:

int a=10,b=20;//declaring 2 variable a,b. Integer data type can have integer value from -32768 to 32767.

float f=20.8; // Float data type must have decimal value

char c='A'; // Character data type can have only one character / alphabet/ digit within inverted comma. ‘5’ or ‘=‘

Data Type

Data type	Size	Range	Format specifier
char	1 byte	-127 to +127	%c
unsigned char	1 byte	0 to 255	%c
short	2 byte	-32,767 to 32,767	%hi, %d, %i
unsigned short	2 byte	0 to 65,535	%hu, %d, %i
int	4 bytes	-2,147,483,847 to +2,147,483,847	%d, %i, %u
unsigned int	4 bytes	0 to 4,294,967,295	%u
long	4 bytes	-2,147,483,847 to +2,147,483,847	%d, %i, %u, %ld, %l, %li
unsigned long	4 bytes	0 to 4,294,967,295	%lu, %u
long long	8 bytes	-9,223,372,036,854,775,807 to +9,223,372,036,854,775,807	%lli
unsigned long long	8 bytes	0 to 18,446,744,073,709,551,615	%llu
float	4 bytes	1.2E-38 to 3.4E+38	%f
double	8 bytes	2.3E-308 to 1.7E+308	%lf
long double	10 bytes	3.4E-4932 to 1.1E+4932	%Lf

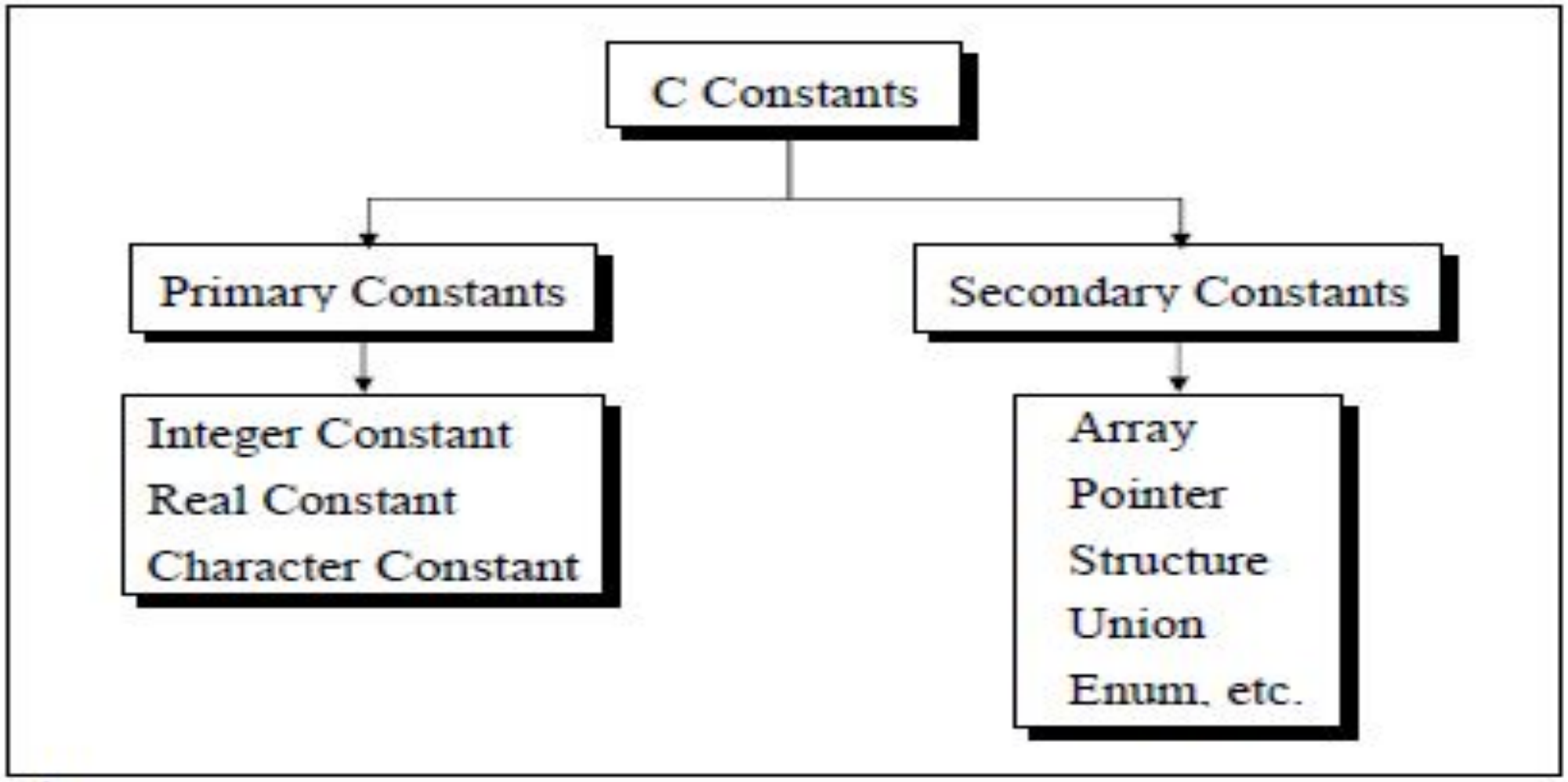
```
// Create variables

int myNum = 5;              // Integer (whole number)
float myFloatNum = 5.99;   // Floating point number
char myLetter = 'D';        // Character

// Print variables


printf("%d\n", myNum);
printf("%f\n", myFloatNum);
printf("%c\n", myLetter);
```

Constant


C Constants
Primary Constants
Secondary Constants
Integer Constant
Real Constant
Character Constant
Array
Pointer
Structure
Union
Enum, etc.

Programming Error

Programming errors are also known as the bugs or faults, and the process of removing these bugs is known as debugging.

There are mainly five types of errors exist in C programming:

- Syntax error
- Run-time error
- Linker error
- Logical error
- Semantic error

Compile Time Vs Run Time

Compile-time	Runtime
The compile-time errors are the errors which are produced at the compile-time, and they are detected by the compiler.	The runtime errors are the errors which are not generated by the compiler and produce an unpredictable result at the execution time.
In this case, the compiler prevents the code from execution if it detects an error in the program.	In this case, the compiler does not detect the error, so it cannot prevent the code from the execution.
It contains the syntax and semantic errors such as missing semicolon at the end of the statement.	It contains the errors such as division by zero, determining the square root of a negative number.

Compile Time Error

```
#include <stdio.h>
int main()
{
 int a=20;
 printf("The value of a is : %d",a):
 return 0;
 }
```

In the above code, we have tried to print the value of 'a', but it throws an error. We put the colon at the end of the statement instead of a semicolon, so this code generates a compile-time error.

Run Time Error

```
#include <stdio.h>
int main()
{
   int a=20;
   int b=a/0; // division by zero
   printf("The value of b is : %d",b);
   return 0;
}
```

In the above code, we try to divide the value of 'b' by zero, and this throws a runtime error.

Printf & Scanf

printf() function: The **printf() function** is used for output. It prints the given statement to the console.

syntax of printf() function:

printf("format string",argument_list);

scanf() function: The **scanf() function** is used for input. It reads the input data from the console.

syntax of scanf() function:

scanf("format string",argument_list);

```
#include<stdio.h>
int main()
{
int number;
scanf("%d",&number);
printf("cube of number is:%d ",number*number*number);
return 0;
}
```

Common Things

Know it:

- **#include <stdio.h>** is a pre-processor directive. When the above program is compiled the compiler replaces the first line with entire contents of stdio.h file. The stdio.h (Standard Input/Output) is called as header file and contains functions for basic input/output operations.
- **int main()** is a function and defines the starting point of our program. **int**: It is the return type of the function. It specifies that the main() function should return an integer value when its work is finished. **main():** It is an identifier or name of the function. Note: The starting point of any program in C is defined by main function.
- **return 0;** returns an integer value to the operating system or the Runtime environment of C acknowledging OS/C Runtime Environment that the program has terminated successfully.

- /* */ is used for Comment.

  Example:

  /* formula for simple interest */

- \n is used for new line

  Example:

  printf("cube of number is \n ",number*number*number);

- **Ctrl + F9** to compile and execute the program
- F2 is used for show the errors Use.
- Use **Alt + F5** to view the output console

First C Program

```
#include <stdio.h>
int main()
{
printf("Hello C Language");
return 0;
}
```

Program to print sum of 2 numbers

```
#include<stdio.h>
int main()
{
int x=0,y=0,result=0;
printf("enter first number:");
scanf("%d",&x);
printf("enter second number:");
scanf("%d",&y);
result=x+y;
printf("sum of 2 numbers:%d ",result);
return 0;
}
```