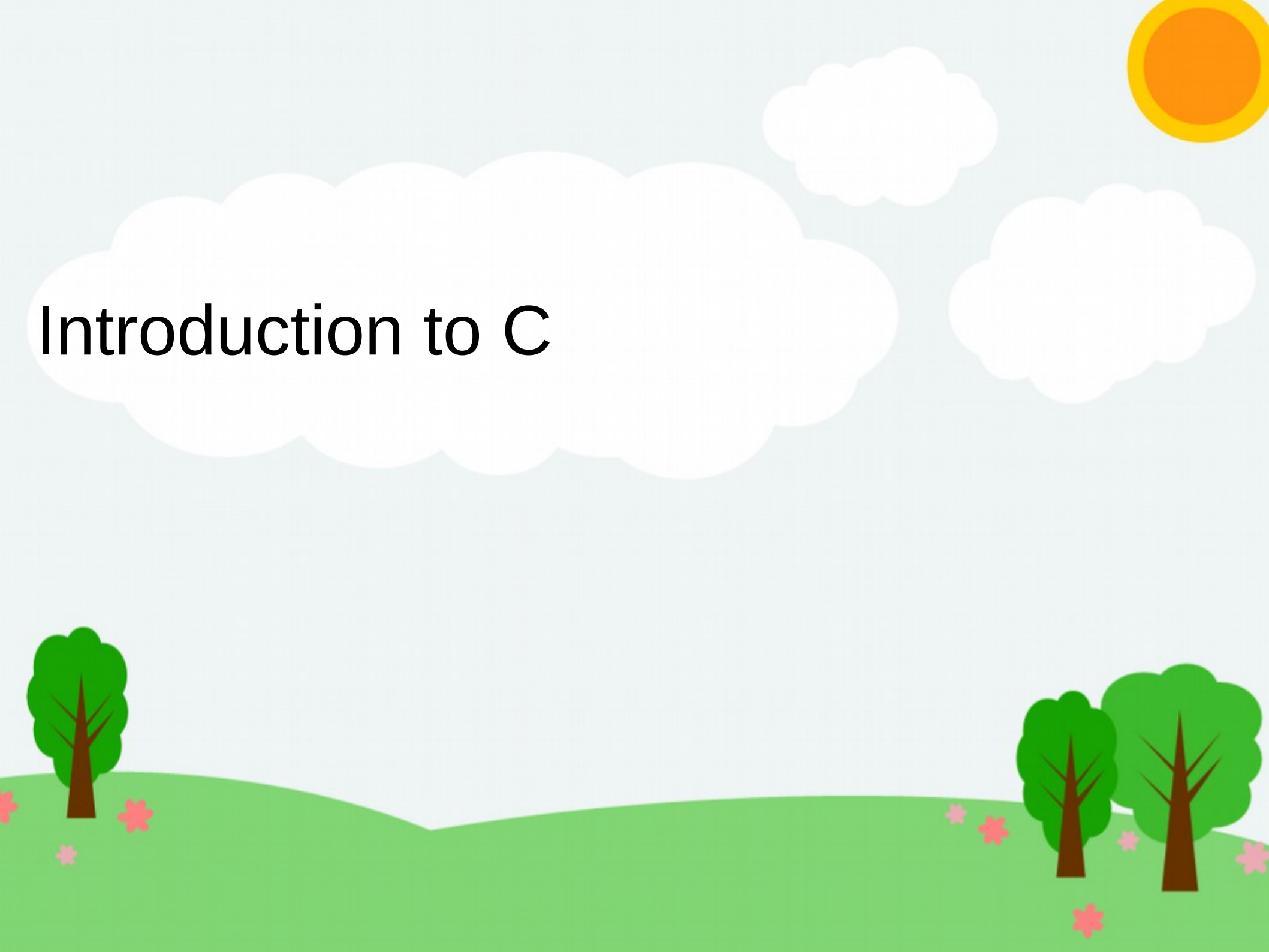
Introduction to C

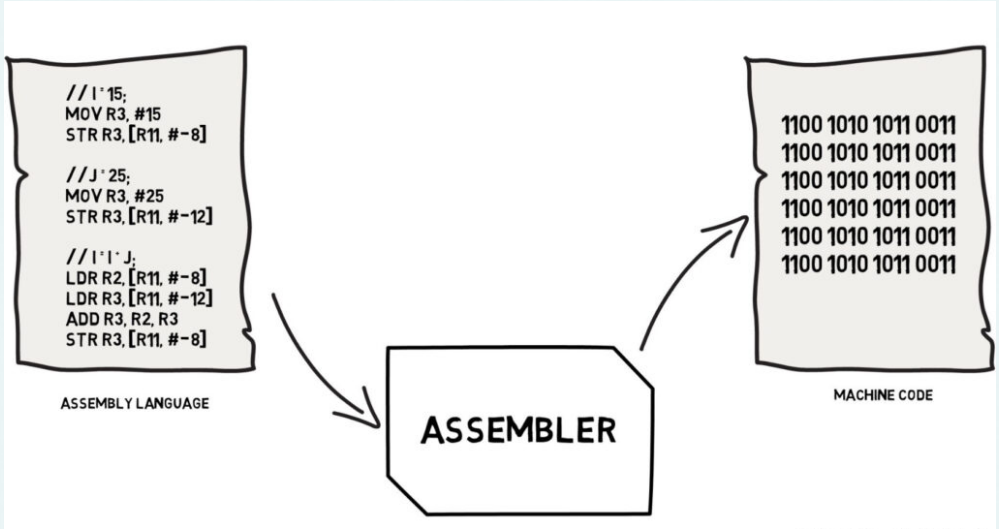


A program

- A set of instructions that are grouped together to accomplish a well-defined task.
- Why programs for computer??
  - Computers has ability to execute programs in order to perform a given task.
- Computer programming (programming or coding) is the process of writing, testing, debugging/ troubleshooting, and maintaining the source code of computer programs
- We take help of a programming language to write programs
- A programming language is a machine-readable artificial language designed to express computations that can be performed by a machine, particularly a computer

Broad types of programming language

- **Low level language**
  - **Machine level language** : It is a collection of bits to be read and interpreted by a computer. It is the only language understood by a computer.
  - **Assembly level language** : symbolic representation of machine language with a high level of correspondence between the language instructions and the machine code instructions of the architecture.
- **High level language**
  - Designed keeping in mind features of portability
  - Machine independent
  - Easy to write and understand
  - The programmer pays whole attention logic of the program
  - Eg. C, C++, JAVA, FORTRAN, COBOL, ...



Translators

- For translating high level and low level language to machine language
- Assembler: Assembly Level to Machine Level
- Interpreter : High Level to Machine Level
  - Interpreter searches the error statement by statement
- Compiler : High Level to Machine Level
  - Compiler searches all errors in the code and lists them
- Which programming language has both compiler and interpreter?

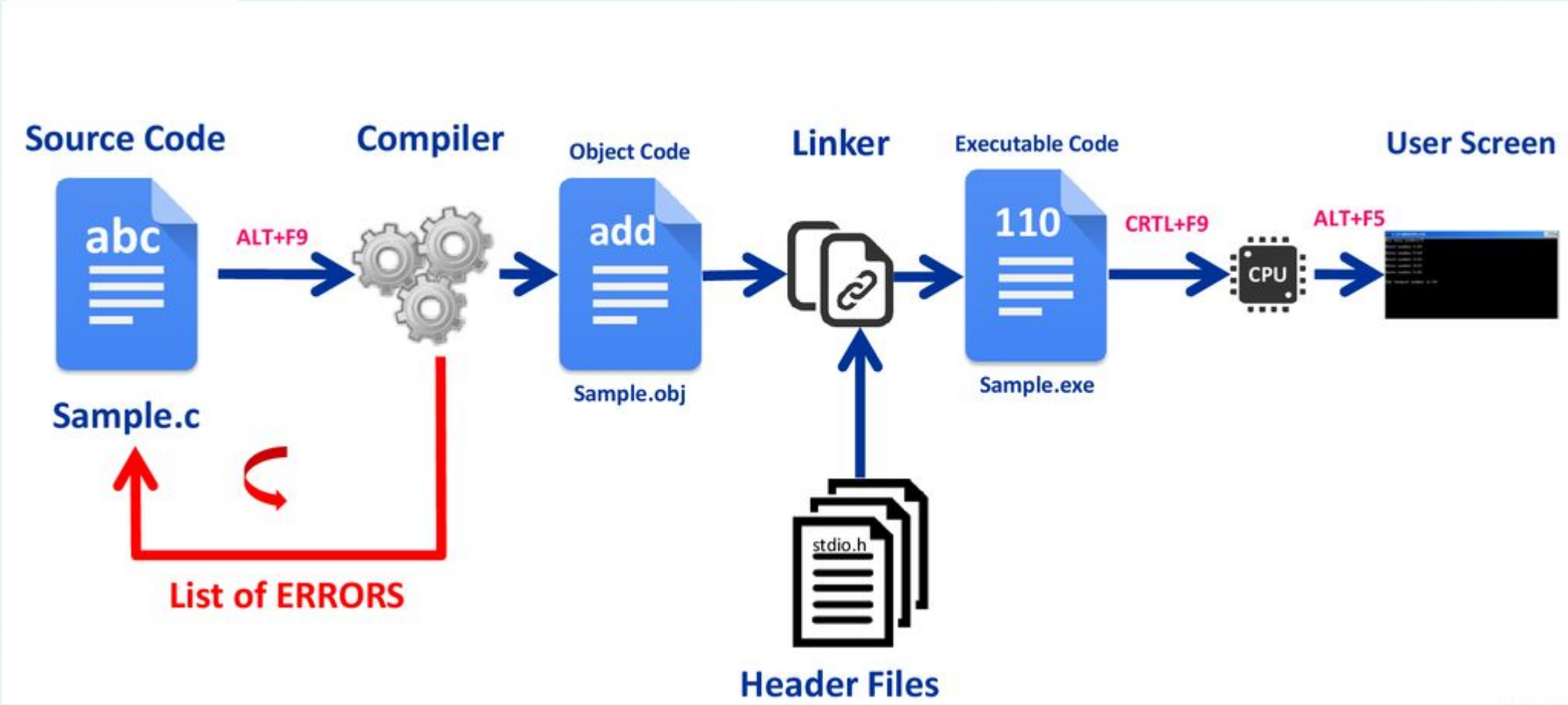
C programming language

- Initially developed by Dennis Ritchie in the year 1972 at Bell Laboratories of AT&T Labs
- BCPL -> B -> C

Why C?

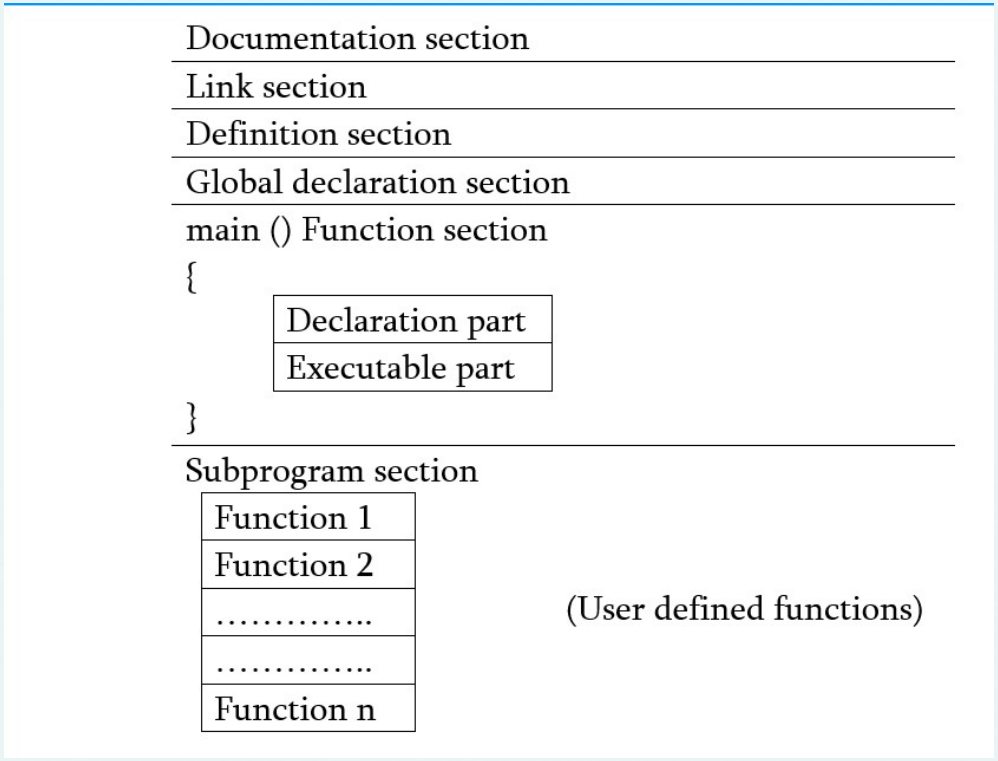
- General purpose
  - A basic foundation for learning programming language elements
- Major parts of popular OS like windows, Linux,Unix are written in C
- Embedded System Programs are written in C
- Low-level Memory Access : C provides several language elements that makes interaction with hardware
- C is Faster compared to other programming laguage

C program execution process



Generic structure of C program

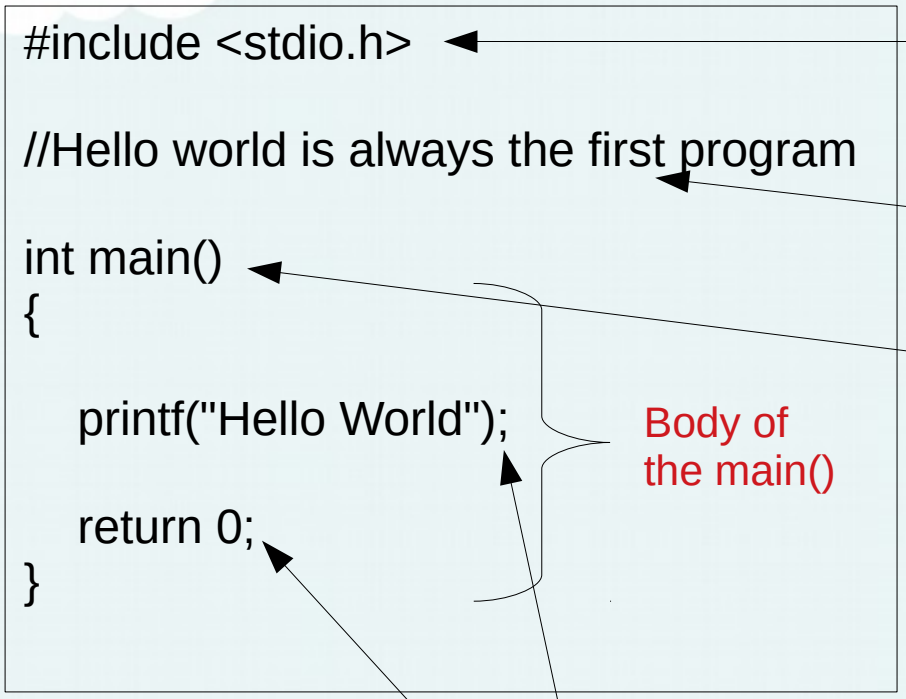
- A C program, irrespective its size, consists of functions and variables.
- A function contains statements that specify the computing operations to be done, and variables store values used during the computation



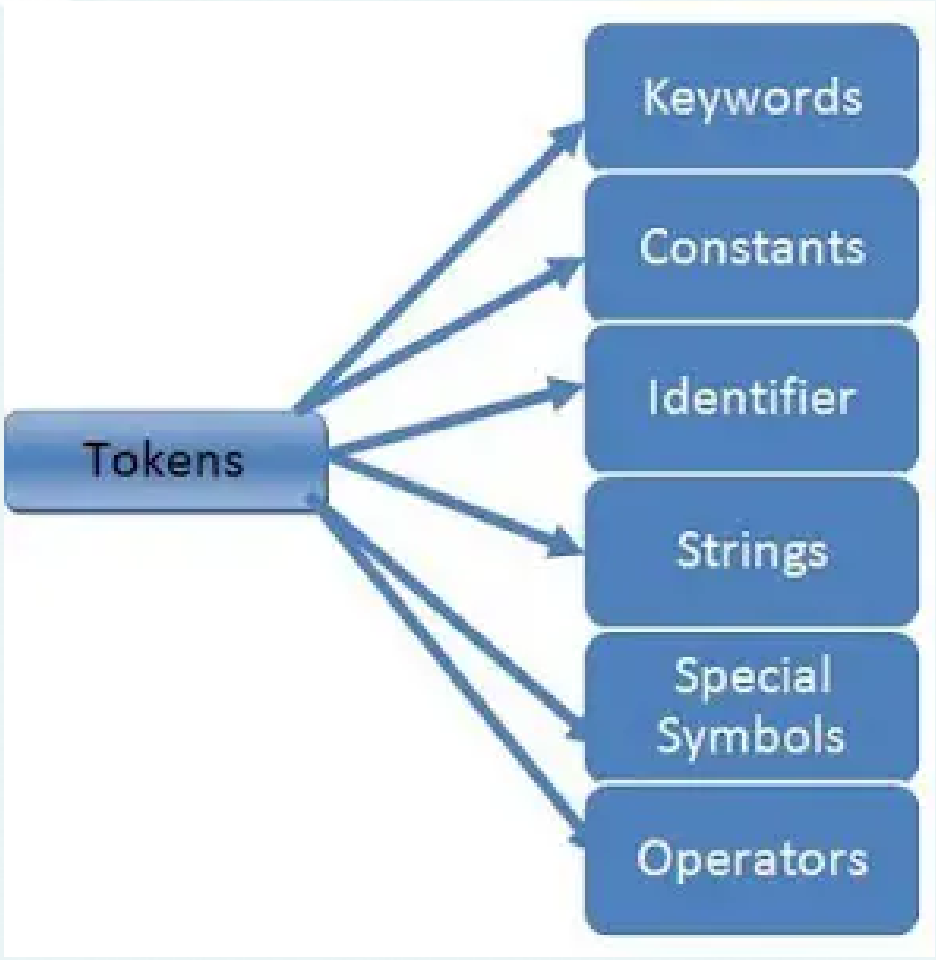
C language standards

- most commonly used are
- C89/C90 (ANSI C or ISO C) was the first standardized version of the language, released in 1989 and 1990, respectively.
- C99 (ISO/IEC 9899:1999) introduced several new features, including variable-length arrays, flexible array members, complex numbers, inline functions, and designated initializers. This standard also includes several new library functions and updates to existing ones.
- C11 (ISO/IEC 9899:2011) introduced several new features, including _Generic, static_assert, and the atomic type qualifier. This standard also includes several updates to the library, including new functions for math, threads, and memory manipulation.
- C18 (ISO/IEC 9899:2018) is the most recent standard

The hello world program



C tokens



- A token is defined as the smallest individual element of a programming language that is meaningful to the compiler
- C programming language has six types of token

Keywords

- The keywords are pre-defined or reserved words
- Each keyword is has a specific function/meaning in a program
- They can’t be used as variable names as we are not allowed to provide a new meaning to it
- However, we can create alias for a keyword by using C preprocessor directives
- C has 32 keywords
- Must be written in lowercase letters

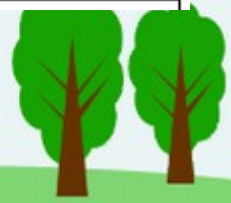
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Identifiers

- Identifiers are used as the general terminology for the naming of variables, functions, arrays, and structures
- These are user-defined names consisting of an arbitrarily long sequence of letters and digits
- We can create an identifier by declaration and can use it in later program statements to refer to the associated value
- The **rules** to define an Identifier
  - Must begin with a letter or underscore(_).
  - The name should consist of only alphabets (A,B.....Z or a,b.....z), digits (0,1.....9), or underscore symbol (_)
  - No other special character is allowed.
  - It should not be a keyword.
  - It must not contain white space.
  - Most of the compiler support length upto 31 characters

Identifiers(2)

Identifiers	Valid / Invalid	Give Reasons if INVALID
(a) record1	valid	
(b) 1record	invalid	Can't start with a number
(c) file_3	valid	
(d) return	valid	
(e) $tax	invalid	Must start with a letter
(f) name	valid	
(g) name and address	invalid	Spaces can't be used in identifiers
(h) name_and_address	valid	
(i) name-and-address	invalid	The (-) symbol can't be used in identifiers

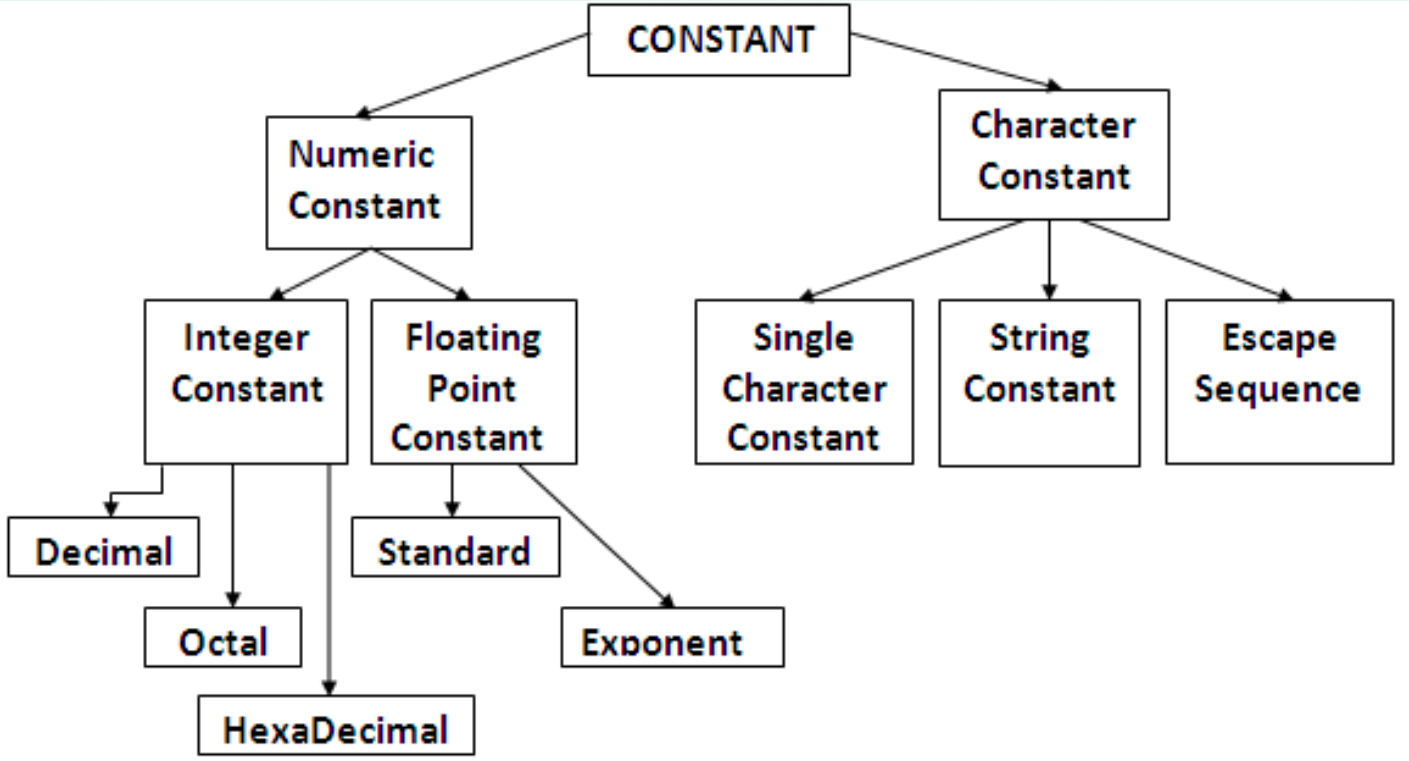


Variable in C

- Variable is a name used to store values
- Values assigned to variable can be changed during execution of a program
- Variables are associated with a data type indicating what type of data it holds
- A variable occupies memory during execution

Constants

- The constant refers to a fixed values which does not change
- Constants may belong to any of the data types



Examples of constant

- Interger constant : 101 490 4467
- Octal constant : 0123 06663
- Hexadecimal constant : 0X123 0X6663
- Floating point constant (standard) :
  - 320.25 -65.50 +3.5 5.450
- Floating point constant(exponent) :12.33E09
- Character constant : ‘a’, ‘A’, ‘3’, ‘\n’, ‘#’

String constant

- It is defined as a sequence of one or more characters enclosed in double quotes (“)
- We can use alphabets, digits, special characters as well as blank spaces in a string constant.
- String constant is always terminated with a special character known as null character (‘\0’).
- This null character indicates the end of the string.
- Example : “nitjsr”, “2024”, $1000”, “alekha.cse@nitjsr.ac.in”, etc.

Special symbols/characters

- The characters other than alphabets
- ` ~ @ ! $ # ^ * % & ( ) [ ] { } < > + = _ – | / \ ; : ' “ , . ?

Operators

- Operators are applied to C variables and other objects to generate new results
- The data used in operation are called operands
- Unary Operators: Those operators that require only a single operand to act upon are known as unary operators
- Binary Operators: Those operators that require two operands to act upon are called binary operators
- Ternary Operator: The operator that requires three operands to act upon is called the ternary operator.
- Also categorized based on the kind of operation performed

Data types

- The way data is represented within computer memory
- Description of the kind of data stored, passed and used.
- Each data type requires different amounts of memory
- They have predefined operations which can be performed over it
- Classes of Data Types
  - Primary (Fundamental) Data Types
  - User-defined Data Types
  - Derived data types
  - Empty Data set

Data Types in C

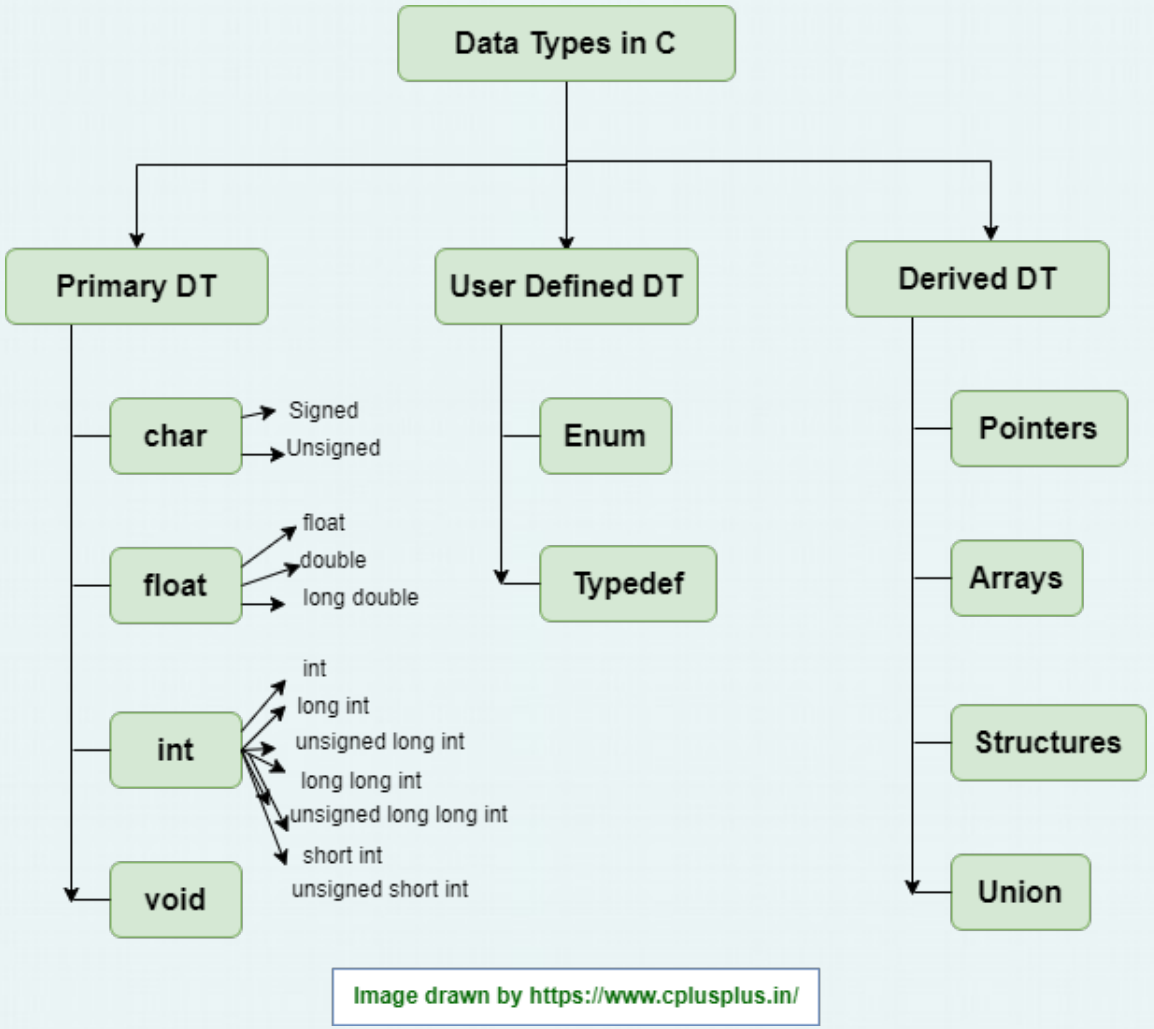


Image drawn by https://www.cplusplus.in/

Use of qualifiers

- Sign Qualifiers
  - signed : for a signed qualifier the number may be positive or negative
  - unsigned : for an unsigned qualifier the number is always positive
- Size qualifier
  - short – smaller variation of the standard data type
  - long – longer variation of the standard data type

Integer data types

- These are used to store integer values, such as whole numbers.
- **short int** – a short integer, typically represented as a 16-bit integer.
- **int** – a standard integer, typically represented as a 32-bit integer.
- **long int** – a long integer, typically represented as a 32-bit integer.
- **long long int** – an even longer integer, typically represented as an 64-bit integer.

Example : integer data type

```
int main()
{
    int intvar1 = 5;
    int intvar2 = -9;
    int intvar3 = 53U;
    long int longvar = 99449L;

    printf("Integer with positive value: %d\n", a);
    printf("Integer with negative value: %d\n", b);
    printf("Integer with unsigned int: %u\n",c);
    printf("Integer with long int : %ld\n", d);

    return 0;
}
```

Character Data Type

- Character data type allows its variable to store only a single character
- Takes 1 byte of memory
- It is the most basic data type in C
- (Signed) chars can range from -128 to 127
- Unsigned chars also store one byte and the possible range of value is 0 to 255

Floating point Data Type

- These are used to store fractional values
- float – a single-precision floating-point number, represented as a 32-bit number.
- double – a double-precision floating-point number, represented as a 64-bit number.
- long double – an extended-precision floating-point number, represented as a 128-bit number.

Example : floating point data type

```
int main()
{
    float fvar1 = 2.5f;

    float fvar2 = 2E-4f;
    printf("%f\n", fvar1);
    printf("%f\n", fvar2);

    double dvar1 = 123123123.00;
    double dvar2 = 457458457.992313;
    printf("first var = %lf\n", dvar1);
    Printf("second var = %lf\n", dvar2);

    return 0;
}
```

Summary of Data types

Data Type	Size (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	16	3.4E-4932 to 1.1E+4932	%Lf

Defining constant in a program

- A constant can be defined using const keyword and a variable
- They are like normal variables but with the difference that their values can not be modified in the program once they are defined.
  - const int c_var = 20;
  - const float fvar = 200.55;
  - const int* const ptr = &c_var;

Symbolic constant

- A symbolic constant is a name that substitute a sequence of characters
- Character may represent a numeric, a character, or a string constant
- Pre-processor directive #define is used for defining symbolic constants.
- The compiler knows that symbolic constants won't change. It substitutes the value for the constant at compile time (preprossessing)
- Syntax
  - #define symbolic_constant_name value_of_the_constant
- Example
  - #define PI 3.141592
  - #define GOLDENRATIO 1.6
  - #define MAX 500
- In a statement like **area = PI * rad * rad;** in a program, the compiler substitutes the name PI by 3.141592

What's Next?
Input and Output in C