# Structure of C Program

- A C program structures basically consists of the following parts:

➢Documentation Section

➢Preprocessor

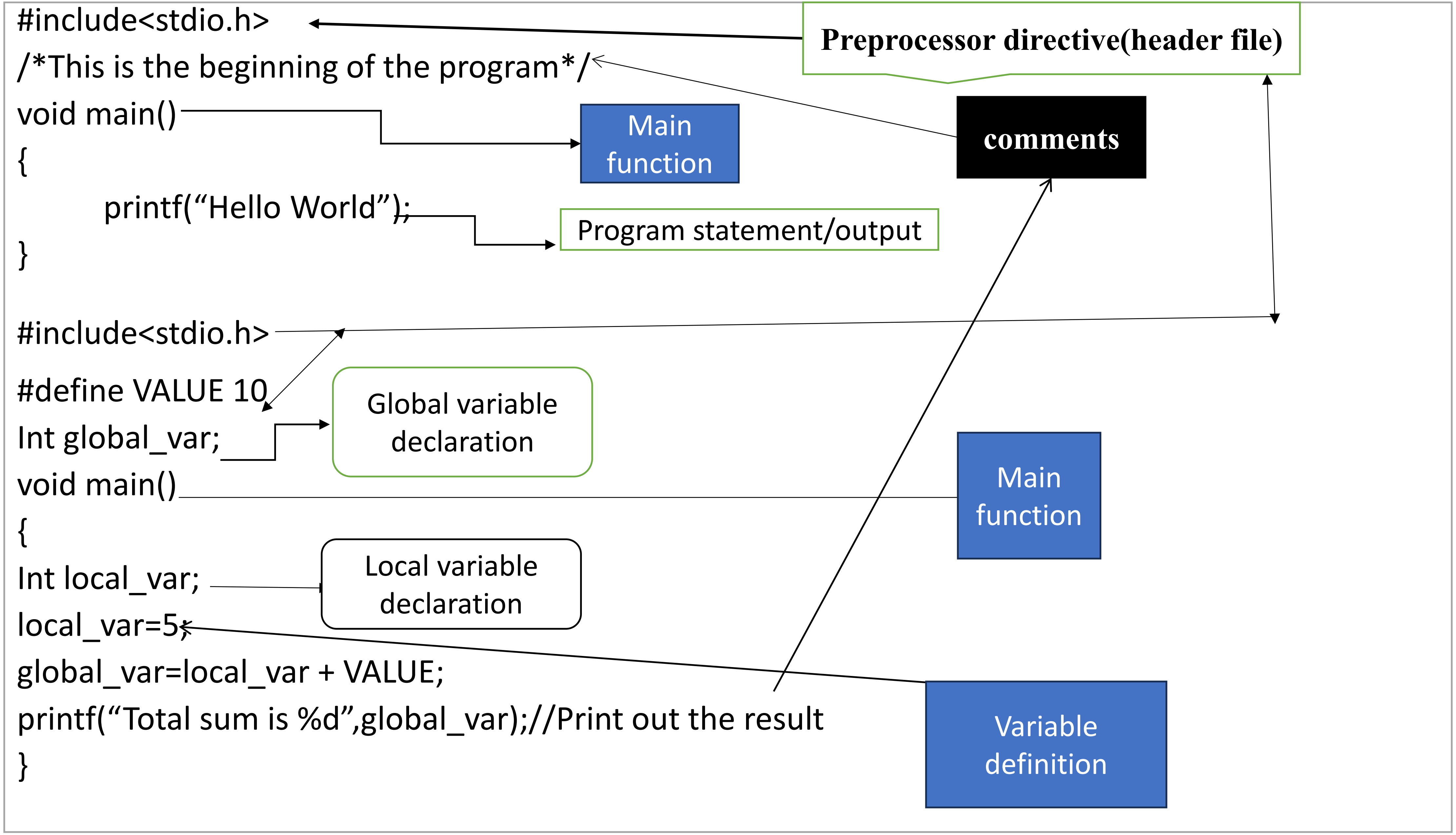
➢Global Variable Declarations Section

➢The main function Section

➢Input and Output Section

➢End of the program

# Conti….



```
#include<stdio.h>
/*This is the beginning of the program*/
void main()
{
        printf("Hello World");
}

#include<stdio.h>
#define VALUE 10
Int global_var;
void main()
{
Int local_var;
local_var=5;
global_var=local_var + VALUE;
printf("Total sum is %d",global_var);//Print out the result
}
```

**Preprocessor directive(header file)**

**comments**

Main function

Program statement/output

Global variable declaration

Main function

Local variable declaration

Variable definition

# Documentation section

**Documentation section:**The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.

- Comments are the way of explaining what makes a program.
- Comments are completely ignored by the compiler during compilation and have no effect on program execution.
- Comment can be used anywhere in the program to add information about program or code, which will be helpful to understand the existing code in the future easily.
- Basically , comments has two types:
  - ▪ **Single line comments**=> starting with '//'
  - ▪ **Multiline comments=>** start with '/* and ends with '*/'

# Preprocessor Directives

**Preprocessors** are the programs that process our source code before the process of compilation.

- This section provides instructions to the compiler to link functions from the system library such as using the *#include directive.*
- The first statement to be checked by the compiler.
- Preprocessor Directives always preceded with ‘#’ sign
- All the header files of the program will be declared in the preprocessor section of the program
- There are a few compiler directives. But only 2 of them will be discussed here.
- – **#include <stdio.h>**
- Tells the compiler to include the file stdio.h during compilation
- Anything in the header file will be included as part of the program
- – **#define VALUE 10**
- Tells the compiler to substitute the word VALUE with 10 during compilation

Conti…

**#include<stdio.h>**

- This is standard Input output header file it allows us to use some commands which includes a file called stdio.h
- This is the preprocessor command. That notifies the compiler to include the header file stdio.h in the program before compiling the source code

#**define**

- The #define preprocessor is used to create a constant throughout the program.
- Whenever this name is encountered by the compiler, it is replaced by the actual piece of defined code.

# Conti…

```
#define PI 3.141592654
main() {
……
perimeter = 2*PI*radius;
area = PI*radius*radius;
…..
}
```

```
main() {
……..
perimeter = 2* 3.141592654 *radius;
area = 3.141592654 *radius*radius;
…..
}
```

- The result of the compilation is the same for both C program (One with #define and the other without it).
- Which one is preferred (less typing)?
- Which one is more readable?
- The one with constant definition using #define preprocessor directive.
- Before compilation, the pre-processor will replace all PI with 3.141592654.

# Conti…

| **Preprocessor** | **Syntax/Description** |
|---|---|
| Macro | **Syntax:** #define This macro defines constant value and can be any of the basic data types. |
| Header file inclusion | **Syntax:** #include <file_name> The source code of the file "file_name" is included in the main program at the specified place. |
| Conditional compilation | **Syntax:** #ifdef, #endif, #if, #else, #ifndefSet of commands are included or excluded insource program before compilation with respect to the condition. |
| Other directives | **Syntax**: #undef, #pragma#undef is used to undefine a defined macro variable. #Pragma is used to call a function before and after main function in a C program. |

# Global Declaration

There are some variables that are used in more than one function.

- Such variables are called global variables and are declared in the global declaration section that is outside of all the functions.
- Variable and function which are declared in this scope can be used anywhere in the program.
- This section also declares all the user-defined functions.

**Example:**

```
Int num=18;
```

# main () function

- Every C program must have a main function.
- The main() function of the program is written in this section.
- Operations like declaration and execution are performed inside the curly braces{….} of the main program.
- The return type of the main() function can be int as well as void too.
- void() main tells the compiler that the program will not return any value.
- The int main() tells the compiler that the program will return an integer value.
- Main function is compulsory for any c program

**Example:**

void main() or int main()

# Input and Output

- **printf()**

➢It is a function in c, which prints text on the screen. Or this is another pre-defined function of C which is used to be displayed text in the screen.

**Example**

printf(“Hello World”);

- **scanf()**

➢This is another pre-defined function of C which is used to access/input the value from keyboard.

**Example:**

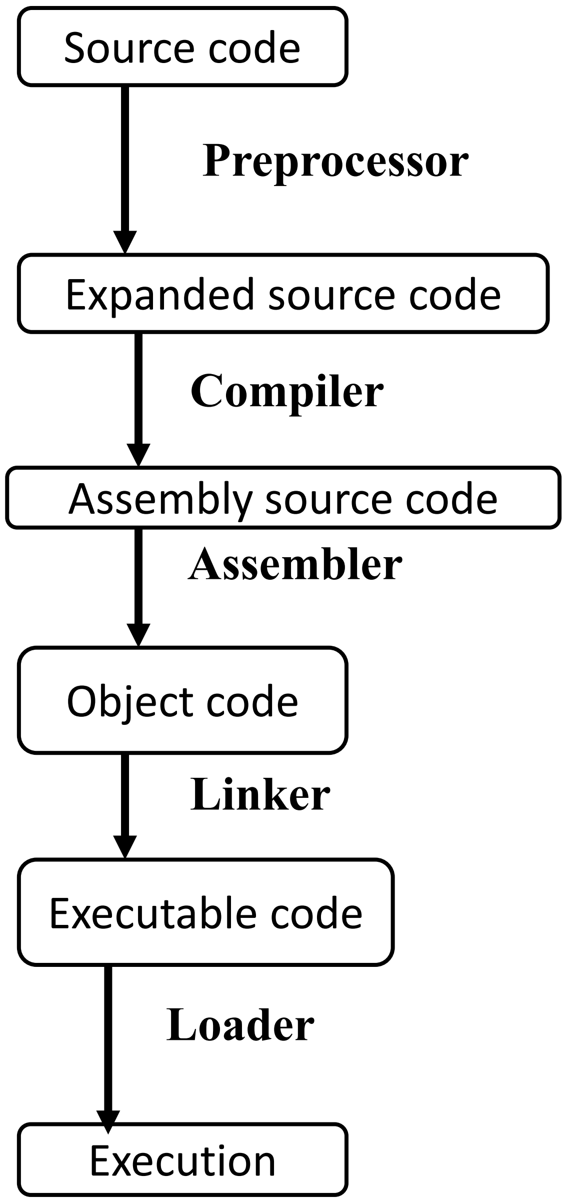
```
int age;
printf("Enter your age: ");
scanf("%d", &age);
```

# Rules in C Programming

- Execution of every C program starts with the main() function.
- Each C program contains only one main() function.
- Every statement in C program ends with semicolon(;).
- C is a case sensitive language. Most of the statement in C program are in lowercase except some symbolic names and outputs.
- Each opening brace should have its closing brace. Braces are used to define the block in coding.
- C can contain comments anywhere in the program.
- The execution of program starts with the opening brace of main() function and ends with the closing brace of the main() function.

# Compilation and Execution of a Program

**Program process flow**



**File name in each steps**

**sample.c**

**sample.i**

**sample.s**

**sample.o**

**sample.exe**

**Description**

- Preprocessor replaces #define (macro),#include (files), conditional compilation codes like #ifdef, #ifndef by their Respective values & source codes in source file
- Compiler compiles expanded source code to assembly source code.
- It is a program that converts assembly source code to object code.
- This is a program that converts object code to executable code and also combines all object codes together.
- Executable code is loaded in CPU and executed loader program