Programming In C

Part I

Introduction to C

Programming Language

- It is used to convert an algorithm into a computer program.
- It has a fixed set of words and rules (syntax and grammer) that are used to write instructions.
- Three types of programming languages:
  - Machine language
  - Assembly language
  - High level language

C Programming

Dennis Ritchie

- It is developed at AT&T Bell laboratories of USA in 1972, designed and written by **Denis Ritchie**.
- At the beginning it was designed for developing the Unix Operating System, then proved itself as powerful general purpose programming language.
- C is a structured programming language and provides modularity.

Why using C?

- C is simple high level language. It contains only 32 keywords
- C programs run faster than programs written in most other languages
- C is the basis for many other languages(Java, C++, C#, and Php...).
- C enables easy communication with computer hardware
- C is portable. Standard compilers exist for virtually every processor.

Developing Programs in C

- There are mainly three steps in developing a program in C:
  - Writing the C program
  - Compiling the program
  - Executing the program

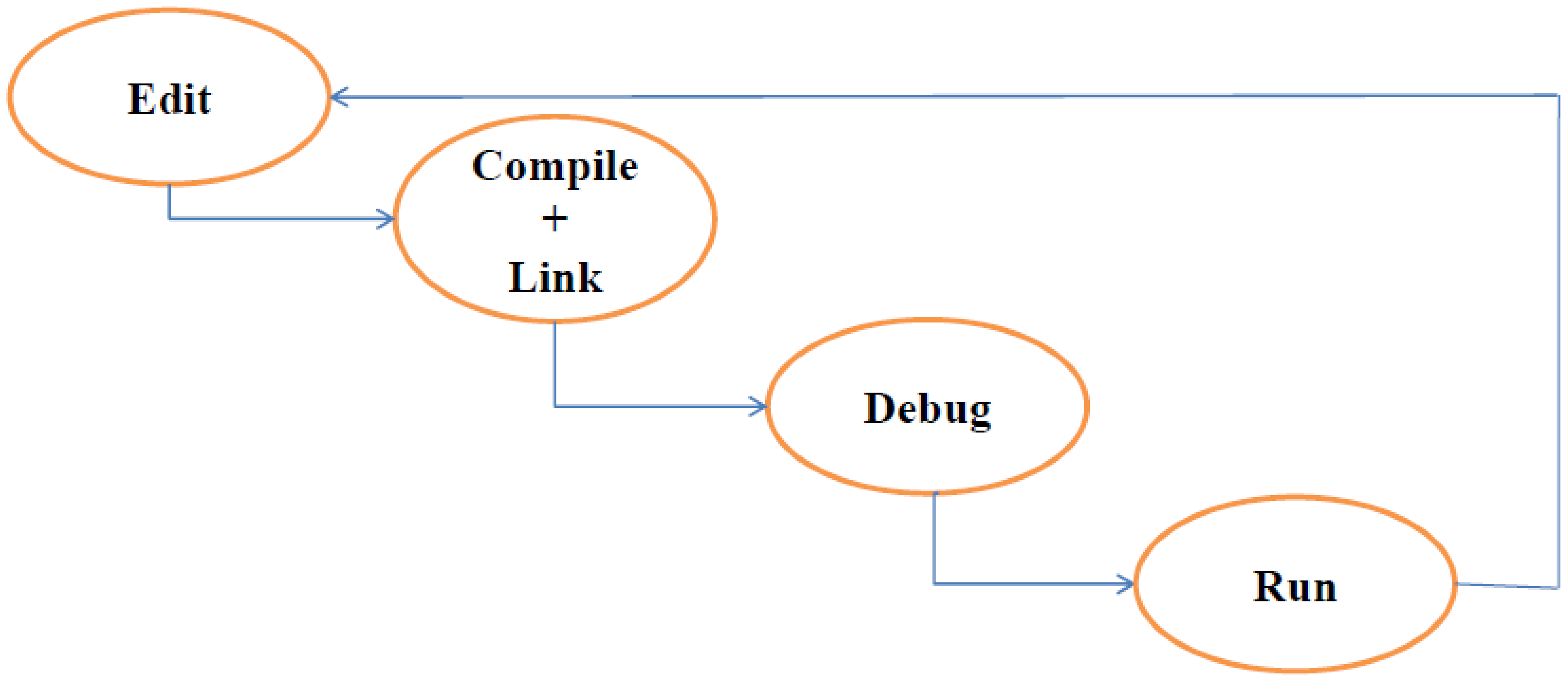
The C Compiler

- The C compiler is a program that converts a C program (referred to as **source code**) into machine language (**object code**).
- A **compiler** generates object code files (machine language) from source code.
- A **linker** combines these object code files into an executable.

Programming Environment

- To facilitate the task of computer programmers, they are provided with a software application know as Itegrated Development Envirement (IDE).
- IDE consists of source code Editor, compiler, and other tools.
- Example of IDEs for C programming: DevC++, CodeBlocks, Eclipse, …etc.

The Programming Process in IDE

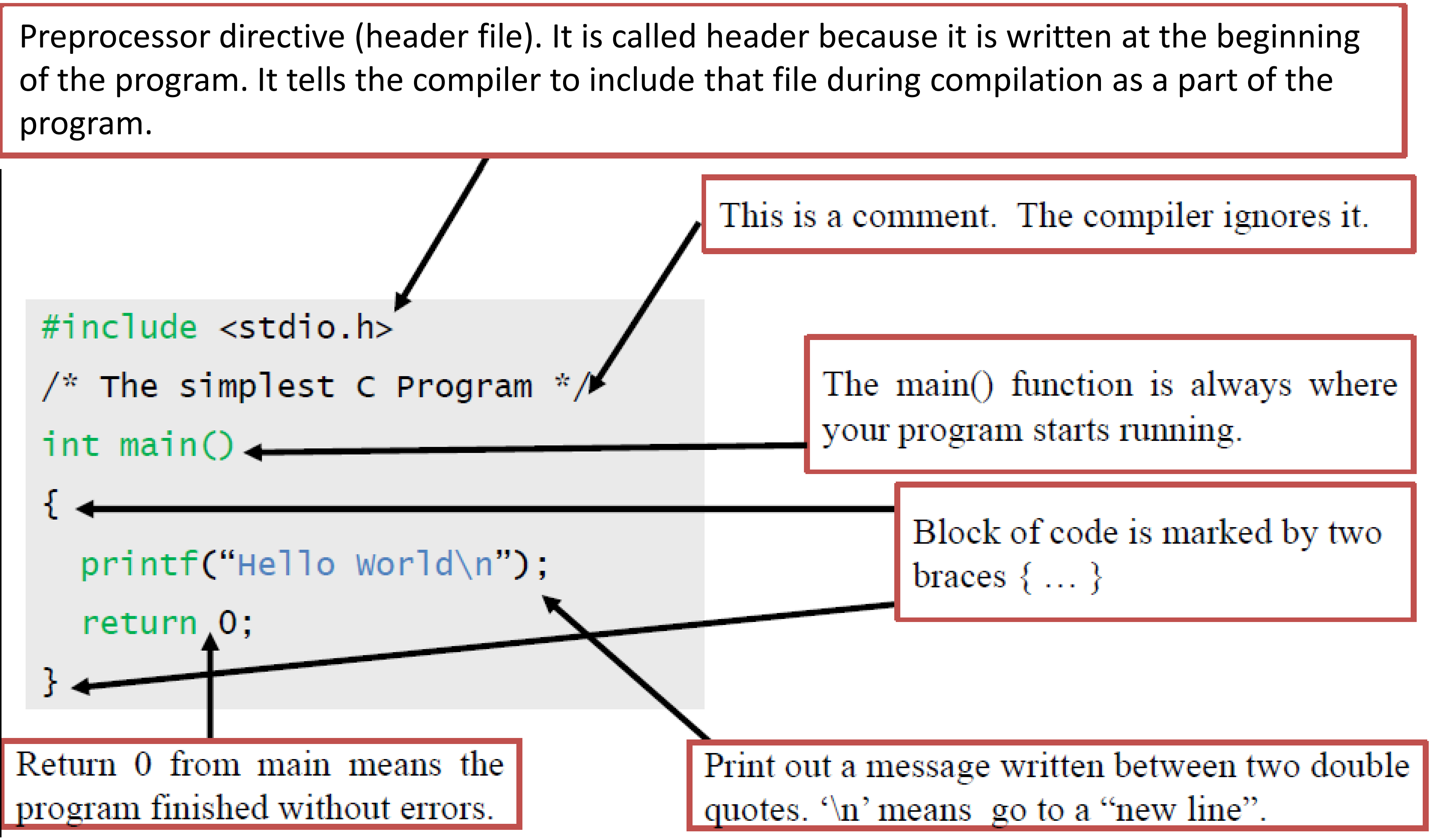


Structure of C Program

- The general structure of a C program is:

```
Preprocessor Directives

int main () {

    Statements

return 0 ;

}
```

First C Program



Preprocessor directive (header file). It is called header because it is written at the beginning of the program. It tells the compiler to include that file during compilation as a part of the program.

This is a comment. The compiler ignores it.

```
#include <stdio.h>
/* The simplest C Program */
int main()
{
  printf("Hello World\n");
  return 0;
}
```

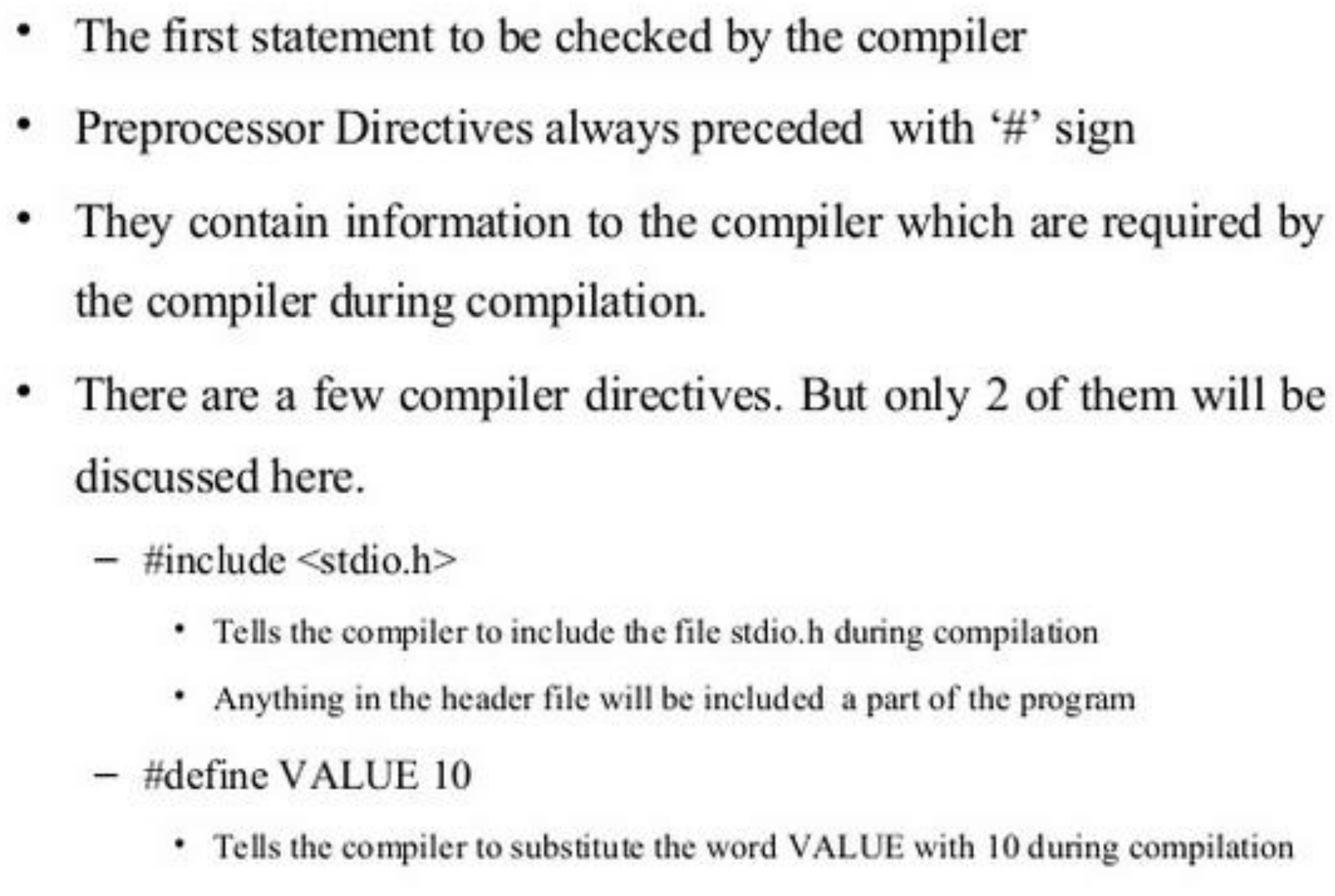
The main() function is always where your program starts running.

Block of code is marked by two braces { … }

Return 0 from main means the program finished without errors.

Print out a message written between two double quotes. '\n' means go to a "new line".

Preprocessor Directives



- The first statement to be checked by the compiler
- Preprocessor Directives always preceded with '#' sign
- They contain information to the compiler which are required by the compiler during compilation.
- 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 a part of the program
  - #define VALUE 10
    - Tells the compiler to substitute the word VALUE with 10 during compilation

Preprocessor Directives (header files)

✓ Header files contain definitions of functions and variables which can be incorporated into any C program by using the pre-processor "# include" statement.

✓ Standard header files are provided with each compiler, and cover a range of areas: string handling, mathematics, data conversion, I\O functions, etc.

Examples of some C header files:

<stdio.h>: STandarD Input\Output library is used for input\output functions. It appears in almost all C source files.

<stdlib.h> STandarD LIBrary declares functions for number conversion, storage allocation, and similar tasks.

<math.h> **:** Includes mathematical functions.

<string.h > **:** Contains all string and bit manipulation functions.

……

Preprocessor Directives (define)

```
#define PI 3.141592654

main() {
    …..
    perimeter = 2*PI*radius;
    area = PI*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)?

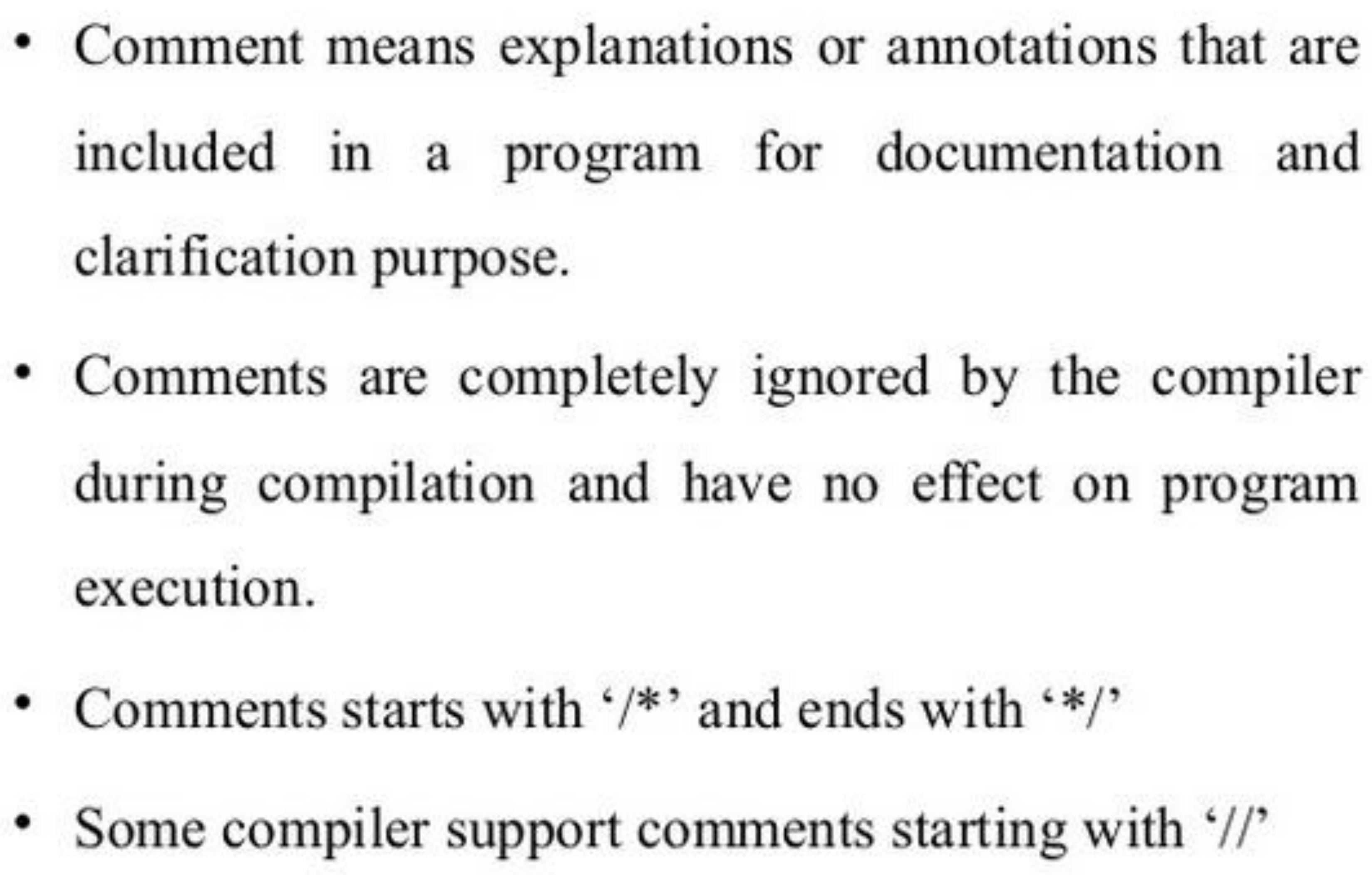
Which one is more readable?

The one with constant definition using #define preprocessor directive.

Before compilation, the pre-processor will replace all PI with 3.141592654.

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

Comments



- Comment means explanations or annotations that are included in a program for documentation and clarification purpose.
- Comments are completely ignored by the compiler during compilation and have no effect on program execution.
- Comments starts with '/*' and ends with '*/'
- Some compiler support comments starting with '//'

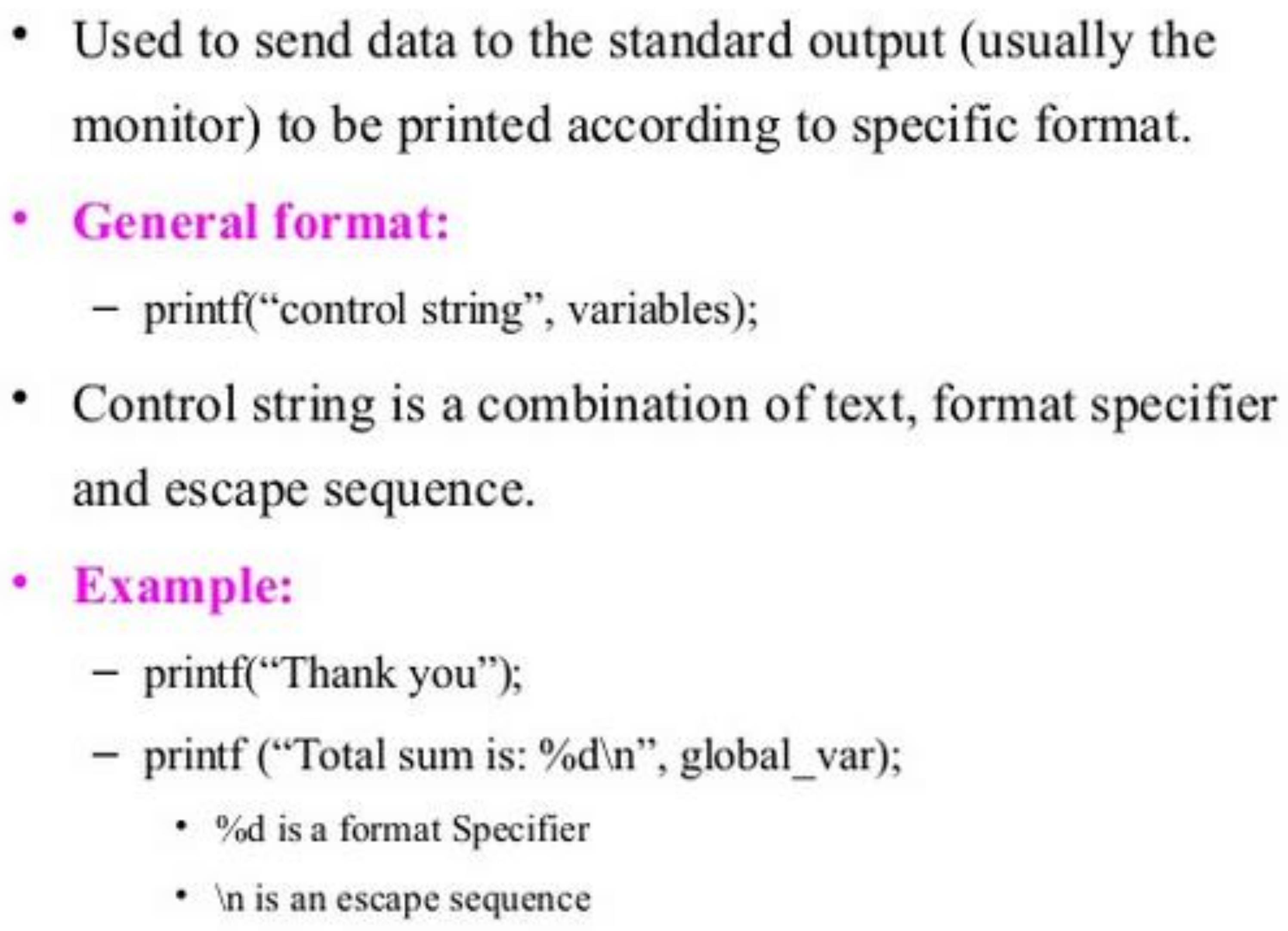
Main Function

- A C program consists of one or more functions that contain a group of statements which perform a specific task.
- A C program must at least have one function: the function **main**.
- We can create our own function or use the functions that has been declared in C library (called Predefined function).
- In order to use Predefined functions, we have to include the appropriate header file (example: stdio.h).

Some functions defined in *stdio.h*

- Two function defined in the header file stdio.h and which are used in almost all C programs are:
  - printf()
  - scanf()

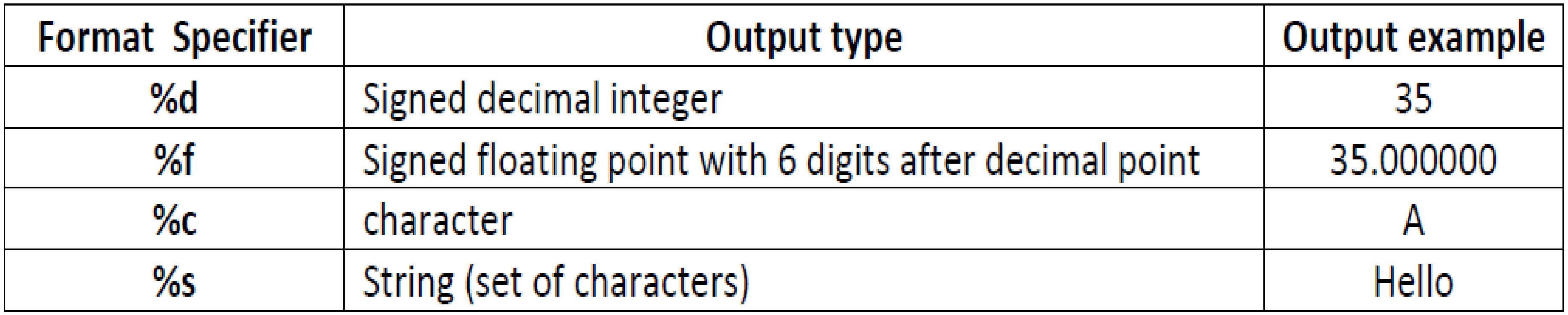
Printf()



- Used to send data to the standard output (usually the monitor) to be printed according to specific format.
- **General format:**
  - printf("control string", variables);
- Control string is a combination of text, format specifier and escape sequence.
- **Example:**
  - printf("Thank you");
  - printf ("Total sum is: %d\n", global_var);
    - %d is a format Specifier
    - \n is an escape sequence

Format specifier

- It tells the printf() function the format of the output to be printed.
- Examples of format specifiers:



Format Specifier	Output type	Output example
%d	Signed decimal integer	35
%f	Signed floating point with 6 digits after decimal point	35.000000
%c	character	A
%s	String (set of characters)	Hello

```c
printf("\n The value of i is %d", i);
```

Here %d is replaced with content of variable i

```c
printf("\n i= %d and j= %d", i,j);
```

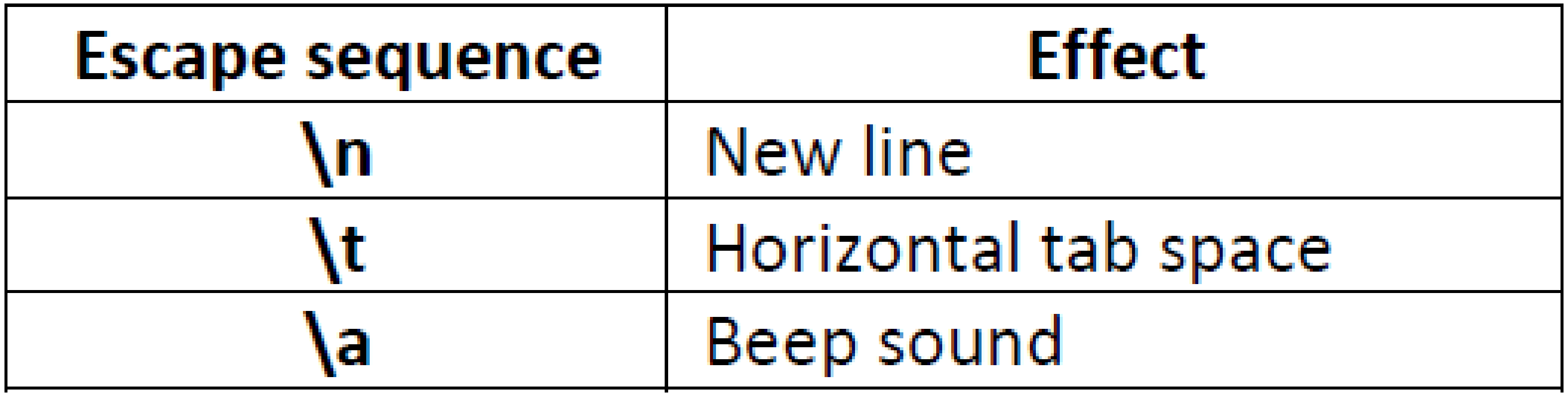
Here first %d is replaced with i and second with j

```c
printf("\n i= %d and j= %d", 7,9);
```

Here first %d is replaced with 7 and second with 9

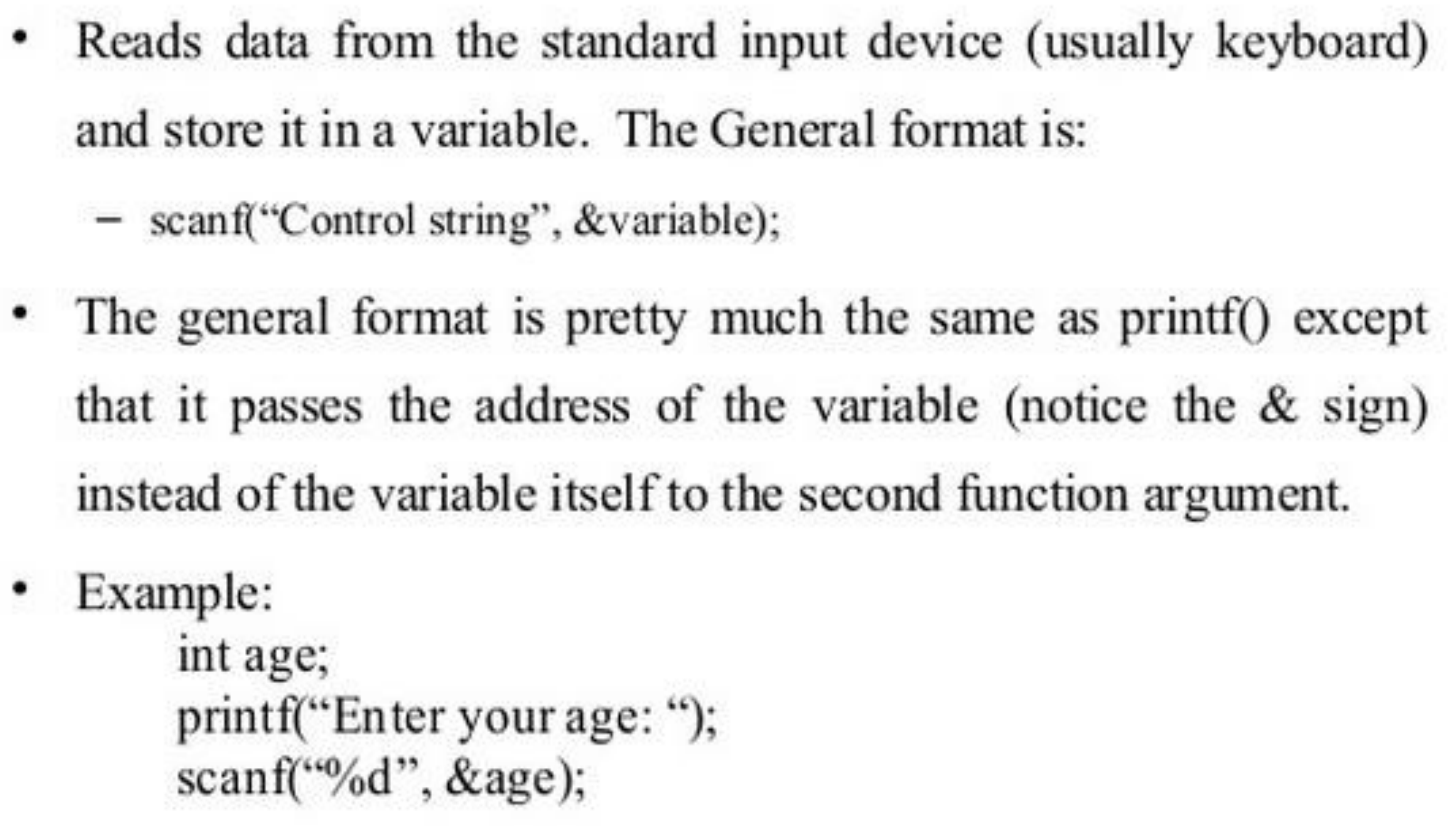
Escape sequence

- It is used in the printf() function to do something to the output.
- Examples:



Escape sequence	Effect
\n	New line
\t	Horizontal tab space
\a	Beep sound

Scanf()



- Reads data from the standard input device (usually keyboard) and store it in a variable. The General format is:
  - scanf("Control string", &variable);
- The general format is pretty much the same as printf() except that it passes the address of the variable (notice the & sign) instead of the variable itself to the second function argument.
- Example:

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

Second C Program

```
#include <stdio.h>

int main()
{
    int x, y, sum;

    printf("Enter the first integer\n");
    scanf("%d",&x);

    printf("Enter the second integer\n");
    scanf("%d",&y);

    sum=x+y;

    printf("The sum is %d\n",sum);

    return 0;
}
```