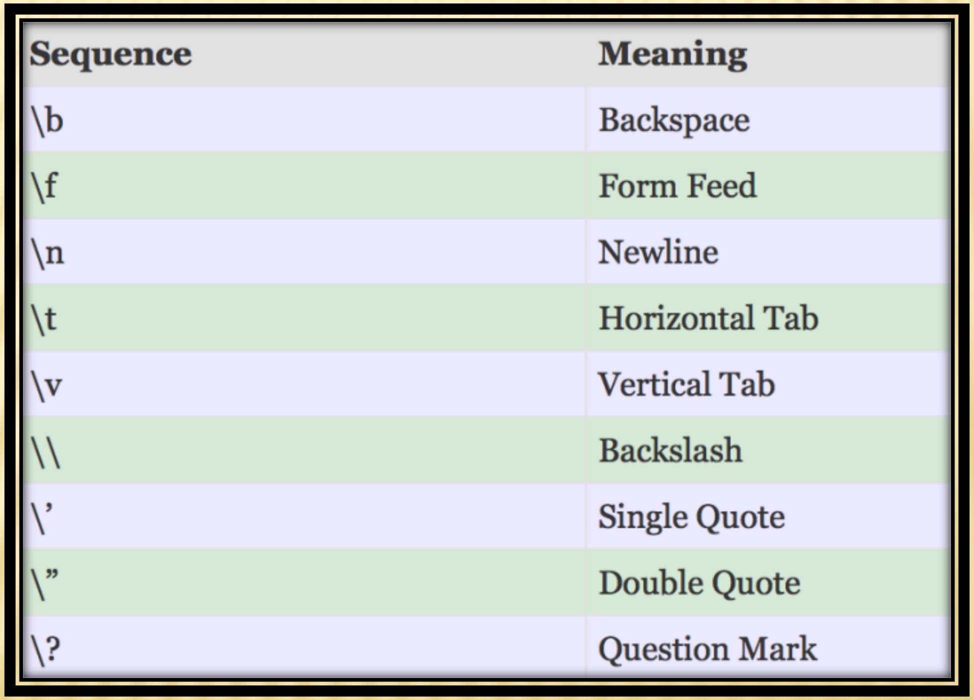
BASIC CONSTRUCTS OF C

FIRST PROGRAM IN C

- Every statement/instruction in C ends with a ;
- *printf* is used to print a string on the output screen
- *#include<stdio.h>* needs to be written at the beginning if we want to use *printf*
- *void main()* is used to enclose the set of executable instructions. It tells us the beginning and end points of the execution via the *{* and *}*

ESCAPE SEQUENCES

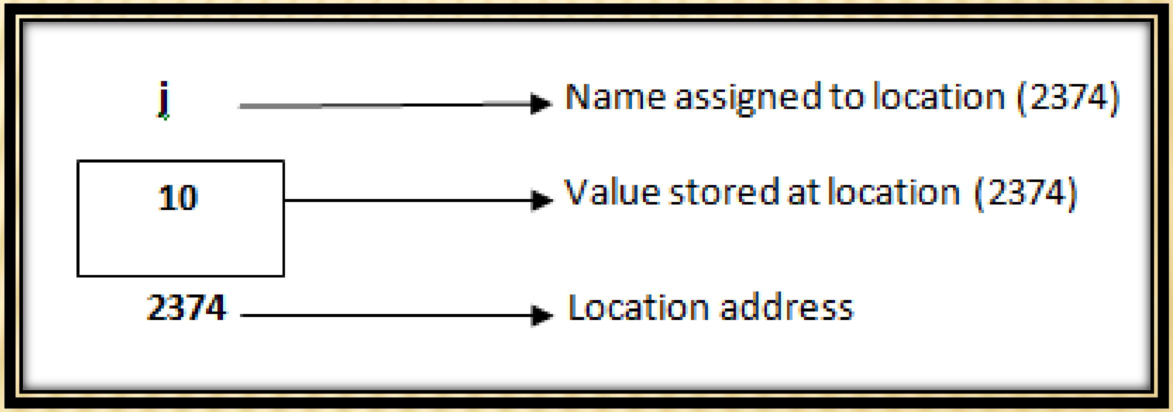
- Special symbols that are used to modify the output in some manner.



Sequence	Meaning
\b	Backspace
\f	Form Feed
\n	Newline
\t	Horizontal Tab
\v	Vertical Tab
\\	Backslash
\’	Single Quote
\”	Double Quote
\?	Question Mark

VARIABLES

- Variables are used to store or hold values in the program.
- They represent a small part of storage in the computer memory (RAM).



i
Name assigned to location (2374)
10
Value stored at location (2374)
2374
Location address

HOW TO USE A VARIABLE?

- Declaration:

  datatype variable_name ;

- Assigning a value to a variable:

  variable_name=value;

- Variable names are called as identifiers
- A variable cannot be used in the program without declaration
- It is advisable to:
  - Give variable names that can easily convey their meaning
  - Keep variable names short and simple

WHAT ARE DATATYPES?

- Data types allow us to specify the type of value that a variable can store.
- There are four main built-in data types:
  - int: Can store only integer values. No fraction/real numbers. Example: 5, 45, -321, 0 etc.
  - float: Can store real numbers with a decimal point. Example: 3.145, -2.5 etc. Can store up to 7 digits after the decimal point.
  - double: Same as float. It has more precision. Can store up to 16 digits after the decimal point.
  - char: Can store any special characters including alphabets and special symbols. Example: ‘A’,’?’,’1’ etc

KEYWORDS

- Keywords are reserved words in C which has a predefined meaning.
- The compiler already knows the meaning of these words.
- It will do a particular operation according to the meaning of the keyword.
- There are 32 keywords in C language.
- Example: void

RULES FOR IDENTIFIERS

- An identifier can be any combination of alphabets, digits and underscore.
- First character should be a letter (alphabet).
- Length of variable name can range from 1 to 8.
- A space in between is not allowed.
- Underscore can be used to concatenate name combinations.
- No commas or other special characters (other than underscore _) are allowed in a variable name.
- C is a case sensitive language – which means a variable name declared as **flag** is not same as **FLAG**.
- You can’t use keywords as identifiers.

QUICK EXERCISE

1. Can you declare a variable that stores the average marks of a student?
2. Can you initialize/assign the above variable a value 67?
3. Can you change the variable value to 50?
4. Can you say what will be the final value of the variable?

HOW TO PRINT A VARIABLE?

- Let’s say we have the following code:

```
int x=5;
printf(“x”);
```

What will be the output?

```
x
```

But we are expecting the value:

```
5
```

HOW TO PRINT A VARIABLE?

- Using printf, we can print any string but we can also print the value of a variable.
- Syntax:

```
printf("format specifier", variable_name);
```

- Format specifiers are characters preceded by a "%"
- They differ based on each datatype

%d-int

%c-char

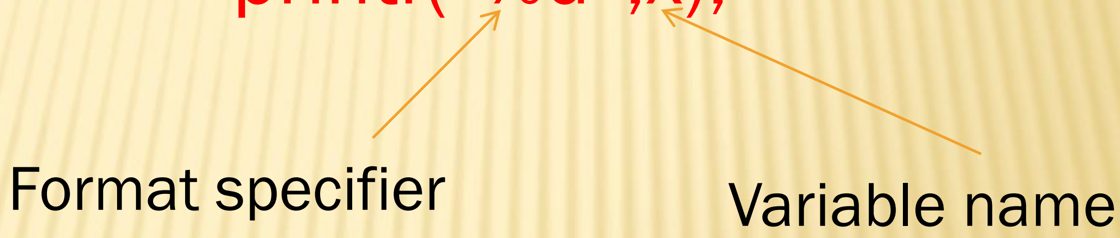
%f-float

%lf-double

HOW TO PRINT A VARIABLE

- Let us come back to the same example:

```
int x=5;
printf("%d",x);
```



Format specifier

Variable name

- The value of the variable "x" is replaced at the format specifier.
- Now the output will be:

```
5
```

HOW TO PRINT MULTIPLE VARIABLES?

- Using printf, we can also print the values of more than one variable.
- Syntax:

  printf("format specifiers", list of variables);

- The variable list is separated with a comma.
- Format specifiers must match the variable.
- Format specifiers are replaced with the variable values from left to right.

HOW TO PRINT MULTIPLE VARIABLES?

- Let us take an example:

```
int x,y,z;
x=5; y=6; z=8;
printf(“%d%d%d”, x,y,z);
```

- What will be the output of the program?

```
568
```

HOW TO PRINT VARIABLES ALONG WITH STRING/TEXT

- In the previous program, the output looks ambiguous/confusing.
- We can also combine any text with variable value.
- Example:

```
int x,y,z;
x=5; y=6; z=8;
printf(“x= %d,y= %d,z= %d”, x,y,z);
```

- Output now will be:

```
x= 5,y= 6,z= 8
```

QUICK EXERCISE

- Can you make modifications to the previous program so that the output is as follows?

```
The value of y is 6
The value of z is 8
The value of x is 5
```

- Use a single “printf” statement only.

LET US SEE HOW TO INPUT A VARIABLE….

HOW TO INPUT A VARIABLE?

- Any program must be as interactive and human friendly as possible.
- Hence, we may want to input a value from the user before the program processes it and gives the output.
- In order to input a variable we use the function "scanf"
- It is quite similar to printf in many ways….

HOW TO INPUT A VARIABLE?

- Syntax of scanf:

```
      scanf(“format specifier”, &variable_name);
```

- Example:

```
 float x;
 scanf(“%f”,&x);
```

- Let us see how this works in the program..

HOW TO INPUT MULTIPLE VARIABLES?

- Syntax:

  scanf(“list of format specifiers”, list of variables with each preceded by a &);

- Example:

  ```
  int x,y,z;
  scanf (“%d%d%d”, &x,&y,&z);
  ```

- <stdio.h> library also includes scanf along with printf

SOME GENERAL POINTS TO NOTE

- Same as printf, format specifiers must match the variables.
- The format specifiers are associated with variables from left to right in scanf also.
- All variable values entered through the keyboard are stored in the variables from left to right
- All input values must be separated by a “space” or “enter” on the output screen.
- Make sure that every input is prompted by a suitable message

QUICK EXERCISE

- Write a program that prompts the user for three integer inputs and print the values that are input by the user….
- Example:

```
Enter three values:
34 45 67          ← These values are entered by the user
The entered values are….
34
45       }        ← These values are printed using printf
67
```