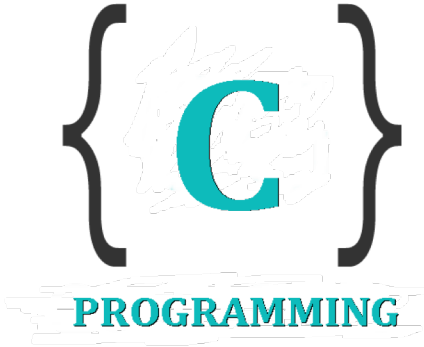
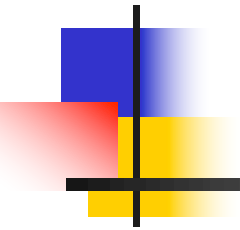


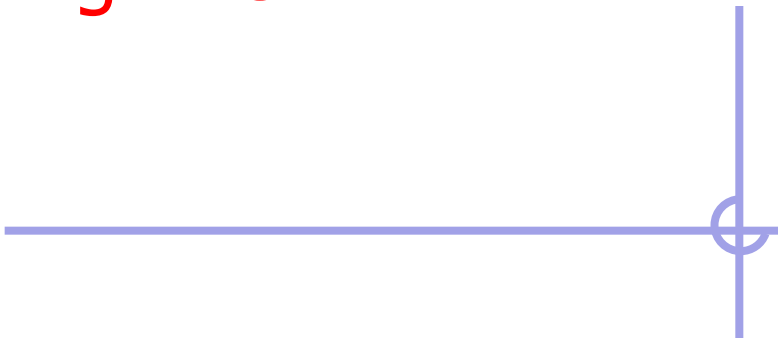
IT 105: Introduction to Programming

Dr. Manish Khare
Dr. Saurabh Tiwari

Lecture 4



Programming in C



History

- Created by Dennis Ritchie at AT&T Labs in 1972
- Originally created to design and support the Unix operating system.
- There are only 32 keywords in the original version of C.
  - for, goto, if, else ……
- Easy to build a compiler for C.
  - Many people have written C compilers
  - C compilers are available for virtually every platform
- In 1983 the American National Standards Institute (ANSI) formed a committee to establish a standard definition.
  - Called ANSI Standard C.
  - As opposed to K&R C (referring to the general "standards" that appeared in the first edition of Brian Kernighan and Ritchie's influential book: *The C Programming Language*)

Why Use C

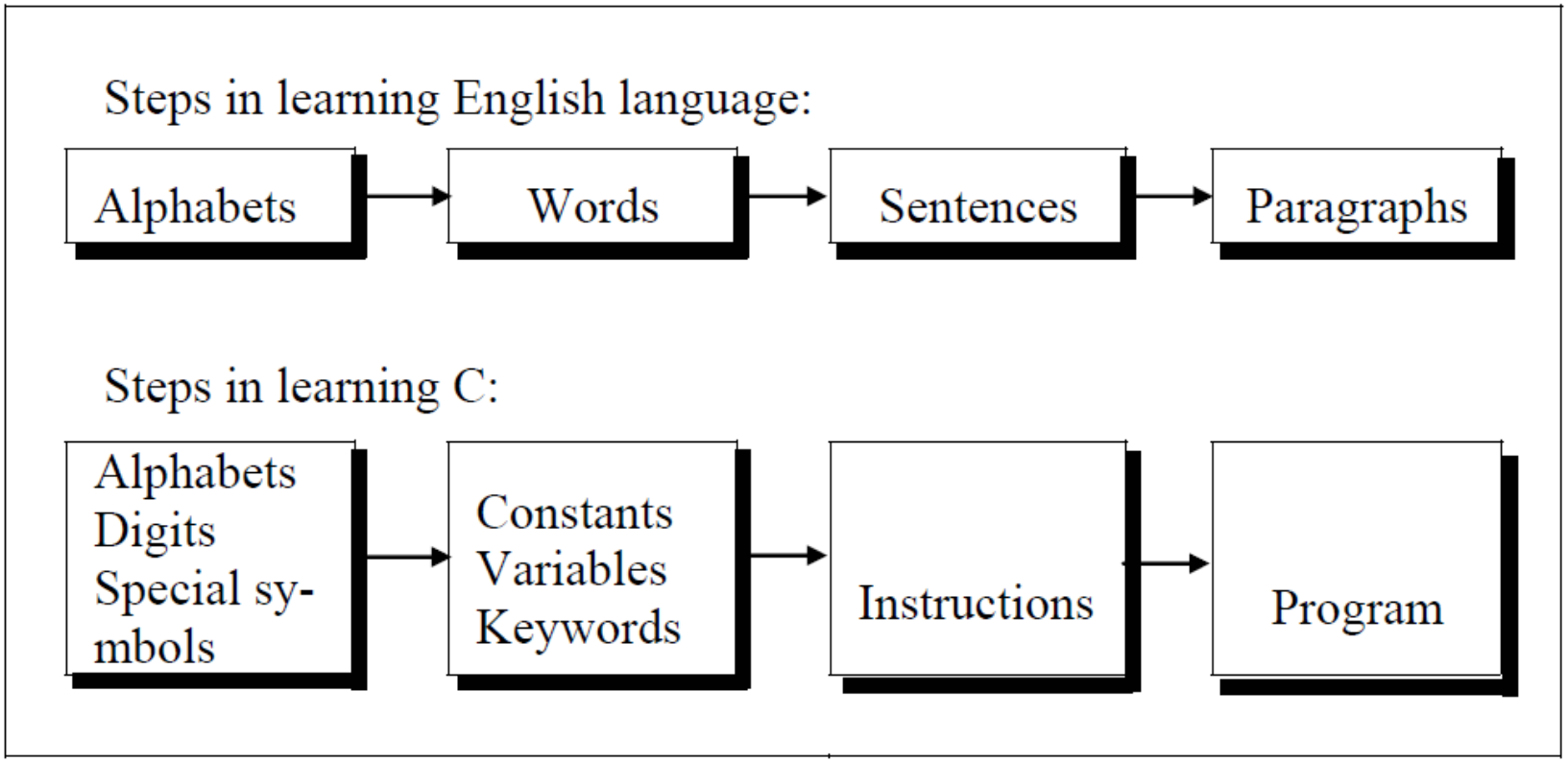
- C is intended as a language for programmers
  - BASIC was for nonprogrammers to program and solve simple problems.
  - C was created, influenced, and field-tested by working programmers.
- C is powerful and efficient
  - You can nearly achieve the efficiency of assembly code.
  - System calls and pointers allow you do most of the things that you can do with an assembly language.
- C is a structured language
  - Code can be written and read much easier.
- C is standardized
  - Your ANSI C program should work with any ANSI C compiler.

C: Characteristics

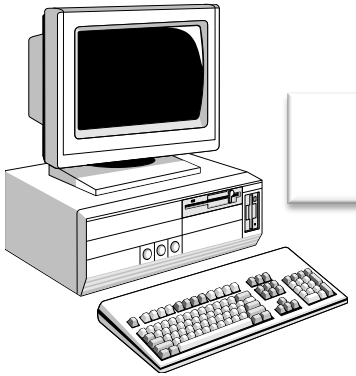
- C takes a middle path between low-level assembly language…
  - Direct access to memory layout through pointer manipulation
  - Concise syntax, small set of keywords
- … and a high-level programming language like Java:
  - Block structure
  - Some encapsulation of code, via functions
  - Type checking (pretty weak)

C: Dangers

- C is not object oriented!
  - Can't "hide" data as "private" or "protected" fields
  - You can follow standards to write C code that looks object-oriented, but you have to be disciplined – will the other people working on your code also be disciplined?
- C has portability issues
  - Low-level "tricks" may make your C code run well on one platform – but the tricks might not work elsewhere
- The compiler and runtime system will rarely stop your C program from doing stupid/bad things
  - Compile-time type checking is weak
  - No run-time checks for array bounds errors, etc. like in Java


Steps in learning English language:
Alphabets
Words
Sentences
Paragraphs
Steps in learning C:
Alphabets
Digits
Special sy-
mbols
Constants
Variables
Keywords
Instructions
Program

C Development Cycle



Edit Program

Source Code

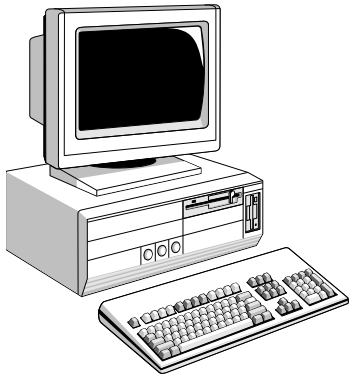
Compile

Object Code

Library Files

Link Object Code

Executable



Structure of a C program

- Every C program consists of one or more functions.
  - One of the functions must be called *main*.
  - The program will always begin by executing the main function.
- Each function must contain:
  - A function *heading*, which consists of the function *name*, followed by an optional list of *arguments* enclosed in parentheses.
  - A list of argument *declarations*.
  - A *compound statement*, which comprises the remainder of the function.

Contd..

- Each compound statement is enclosed within a pair of braces: '{' and '}'
  - The braces may contain combinations of elementary statements and other compound statements.
- Comments may appear anywhere in a program, enclosed within delimiters '/*' and '*/'.
  - Example:
    - a = b + c; /* ADD TWO NUMBERS */

The C Character Set

- A character denotes any alphabet, digit or special symbol used to represent information.

Alphabets	A, B, ….., Y, Z a, b, ……, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /

C Tokens

- When alphabets, digits and special symbols properly combined, then that form a token
- Every C program is a collection of instructions and every instruction is a collection of some individual units.
- Every smallest individual unit of a c program is called token.
- Every instruction in a c program is a collection of tokens.
- Tokens are used to construct c programs and they are said to the basic building blocks of a c program.

- Keywords
- Constants
- Identifiers
- Operators
- Special Symbols
- Strings
- Data values

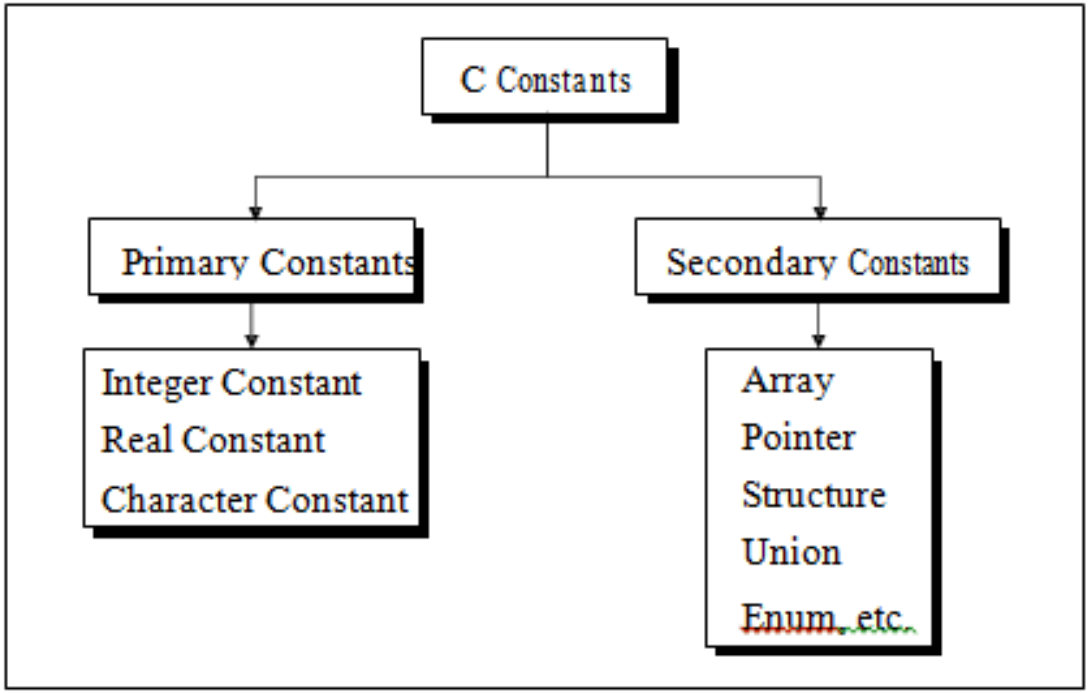
C Keywords

- As every language has words to construct statements, C programming also has words with a specific meaning which are used to construct c program instructions.
- In the C programming language, keywords are special words with predefined meaning.
- Keywords are also known as reserved words in C programming language.
- In C programming language, there are **32 keywords**. All the 32 keywords have their own meaning which is already known to the compiler.

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

C Constants

- In C programming language, a constant is similar to the variable but constant holds only one value during the program execution.
- That means, once a value is assigned to the constant, that value can't be changed during the program execution. Once the value is assigned to the constant, it is fixed throughout the program.
- Type of Constants
  - Primary Constants
  - Secondary Constants



At this time we would restrict our discussion to only primary constants, namely – Integer, Real, and Character constant.

Rules for Constructing Integer Constant

- An integer constant must have at least one digit.
- It must not have a decimal point.
- It can be any of zero, positive or negative.
- If no sign precedes an integer constant, it is assumed to be positive.
- No commas or blanks are allowed within an integer constant.
- The allowable range for integer constants 16 bit is -32768 to 32767.
- -2147483648 to +2147483648 for 32 bit
- Example
  - 426
  - +782
  - -8000
  - -7605

Rules for Constructing Real Constants

- Real constant are often called Floating Point constants. Real constants could be written in two forms
  - Fractional form
  - Exponential form

- Fractional form
  - A real constant must have at least one digit.
  - It must have a decimal point.
  - It could be either positive or negative.
  - Default sign is positive.
  - No commas or blanks are allowed within a real constant.
  - Example
    - +325.34
    - 426.0
    - -32.76
    - -48.5792

Exponential form

- The exponential form is usually used if the value of the constant is either too large or too small.
- In exponential form of representation, the real constant is represented in two parts. The part appearing before ‘e’ is called mantissa, whereas the part following ‘e’ is called exponent
- For ex, 0.000342 can be written in exponential form of 3.42e-4 (which is equivalent to 3.42×10^{-4})

- Exponential form
  - The mantissa part and the exponential part should be separated by a letter e or E.
  - The mantissa part may have a positive or negative sign.
  - Default sign of mantissa part is positive.
  - The exponent must have at least one digit, which must be a positive or negative integer. Default sign is positive.
  - Range of real constants expressed in exponential form is -3.4e38 to 3.4e38
  - Example
    - +3.2e-5
    - 4.1e8
    - -0.2e+3
    - -3.2e-5

Rules for Constructing Character Constants

- A character constant is a single alphabet, a single digit or a single special symbol enclosed within single inverted commas.

- Both the inverted commas should point to the left. For example, ’A’ is a valid character constant whereas ‘A’ is not.

- The maximum length of a character constant can be 1 character.
- Ex.
  - 'A’
  - 'I’
  - '5’
  - '='

String Constants

- Sequence of characters enclosed in double quotes.
  - The characters may be letters, numbers, special characters and blank spaces.
- Examples:
  - “nice”, “Good Morning”, “3+6”, “3”, “C”
- Differences from character constants:
  - ‘C’ and “C” are not equivalent.
  - ‘C’ has an equivalent integer value while “C” does not.

C Identifier and Variables

- Identifiers
  - Names given to various program elements (variables, constants, functions, etc.)
  - May consist of *letters*, *digits* and the *underscore* ('_') character, with no space between.
  - First character must be a letter.
  - An identifier can be arbitrary long.
    - Some C compilers recognize only the first few characters of the name (16 or 31).
  - Case sensitive
    - 'area', 'AREA' and 'Area' are all different.

Valid and Invalid Identifiers

- Valid identifiers
  - X
  - abc
  - simple_interest
  - a123
  - LIST
  - stud_name
  - Empl_1
  - Empl_2
  - avg_empl_salary
- Invalid identifiers
  - 10abc
  - my-name
  - “hello”
  - simple interest
  - (area)
  - %rate

Variables

- It is a data name that can be used to store a data value.
- Unlike constants, a variable may take different values in memory during execution.
- Variable names follow the naming convention for identifiers.
  - Examples :: temp, speed, name2, current

Declaration of Variables

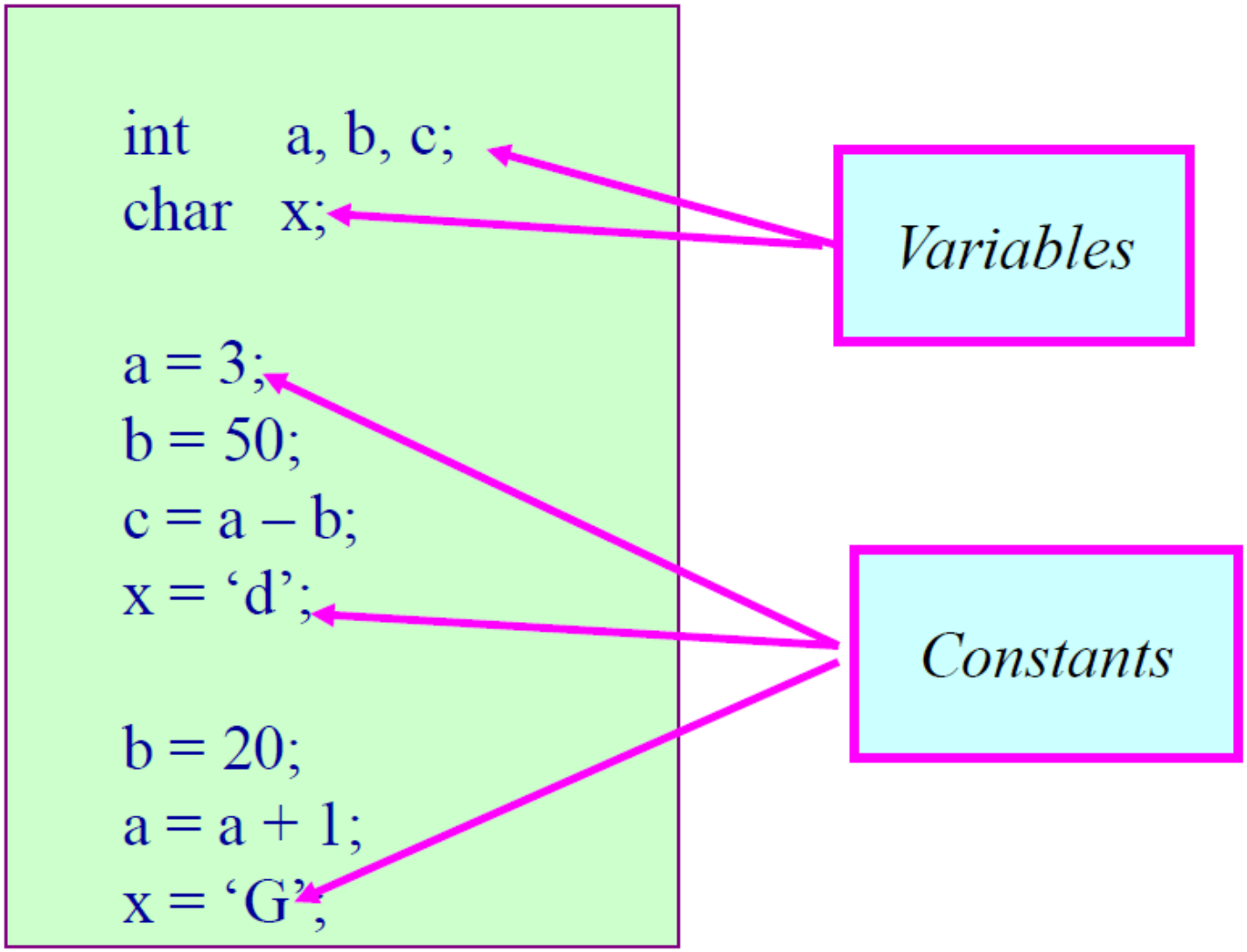
- There are two purposes:
  - 1.It tells the compiler what the variable name is.
  - 2.It specifies what type of data the variable will hold.
- General syntax:
  - data-type variable-list;
- Examples:
  - int velocity, distance;
  - int a, b, c, d;
  - float temp;
  - char flag, option;

Example

```
int     a, b, c;
char    x;

a = 3;
b = 50;
c = a – b;
x = ‘d’;

b = 20;
a = a + 1;
x = ‘G’;
```



- **Valid variable names:**
  - **int** a;
  - **int** _ab;
  - **int** a30;
- **Invalid variable names:**
  - **int** 2;
  - **int** a b;
  - **int long**;

Types of Variables in C

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

Local Variable

- A variable that is declared inside the function or block is called a local variable.
- It must be declared at the start of the block.

```
void function1()
{
int x=10;   //local variable
}
```

- You must have to initialize the local variable before it is used.

Global Variable

- A variable that is declared outside the function or block is called a global variable. Any function can change the value of the global variable. It is available to all the functions.
- It must be declared at the start of the block.

```
int value=20;//global variable


void function1(){
int x=10;//local variable
}
```

Assignment Statement

- Used to assign values to variables, using the assignment operator (=).
- General syntax:
  - variable_name = expression;
- Examples:
  - velocity = 20;
  - b = 15; temp = 12.5;
  - A = A + 10;
  - v = u + f * t;
  - s = u * t + 0.5 * f * t * t;

- A value can be assigned to a variable at the time the variable is declared.
  - int speed = 30;
  - char flag = ‘y’;

- Several variables can be assigned the same value using multiple assignment operators.
  - a = b = c = 5;
  - flag1 = flag2 = ‘y’;
  - speed = flow = 0.0;

Data Types in C

- int:: integer quantity
  - Typically occupies 4 bytes (32 bits) in memory.
- char:: single character
  - Typically occupies 1 byte (8 bits) in memory.
- float:: floating-point number (a number with a decimal point)
  - Typically occupies 4 bytes (32 bits) in memory.
- double:: double-precision floating-point number

- Some of the basic data types can be augmented by using certain data type qualifiers:
  - short
  - long
  - signed
  - unsigned
- Typical examples:
  - short int
  - long int
  - unsigned int

Some Examples of Data Types

- int

  0, 25, -156, 12345, –99820

- char

  'a', 'A', '*', '/', ' '

- float

  23.54, –0.00345, 25.0

  2.5E12, 1.234e-5

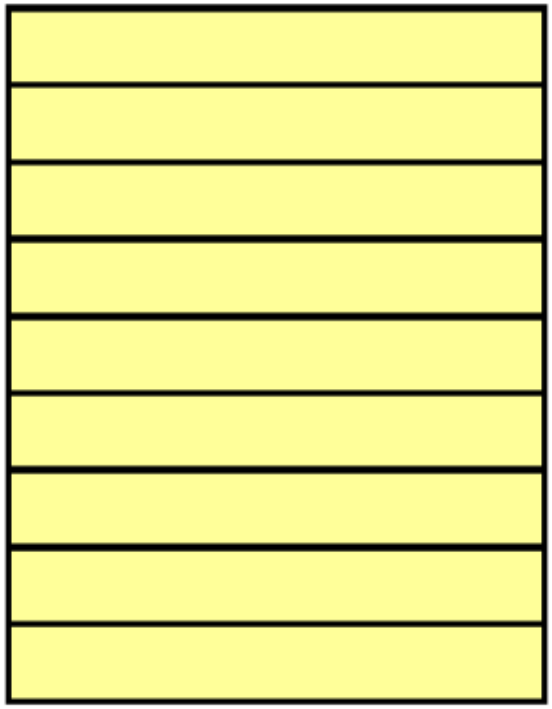
E or e means "10 to the power of"

Memory

- Program stored in Memory?
  - Main Memory

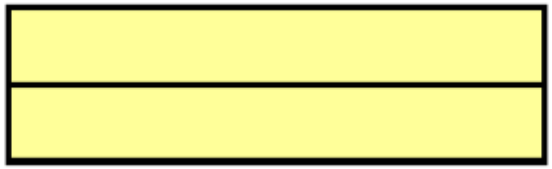
- How does memory look like (logically)?
  - As a list of storage locations, each having a unique address.
  - Variables and constants are stored in these storage locations.
  - Variable is like a *house*, and the name of a variable is like the *address* of the house.
  - Different people may reside in the house, which is like the *contents* of a variable.

Memory Map



Address 0
Address 1
Address 2
Address 3
Address 4
Address 5
Address 6

Every variable is mapped to a particular memory address



Address N-1

Form of a C Program

- Form of a C program indicated how it has to be written.
- There are certain rules about the form of C program that are applicable to all C programs.
  - Each instruction in a C program is written as a separate statement.
  - The statements in a program must appear in the same order in which we wish them to executed.
  - Blank space may be inserted between two words to improve the readability of the statement.
  - All statements should be in lower case letters.
  - C has no specific rules for the position at which a statement is to be written in a given line.
  - Every C statement must be end with a semicolon (;). Thus ; acts as a statement terminator.

Input / Output

- printf
  - Performs output to the standard output device (typically defined to be the screen).
  - It requires a format string in which we can specify:
- The text to be printed out.
- Specifications on how to print the values.
  - printf ("The number is %d.\n", num) ;
- The format specification %d causes the value listed after the format string to be embedded in the output as a decimal number in place of %d.
- Output will appear as: The number is 125

scanf

- Performs input from the standard input device, which is the keyboard by default.
- It requires a format string and a list of variables into which the value received from the input device will be stored.
- It is required to put an ampersand (&) before the names of the variables.
  - scanf ("%d", &size) ;
  - scanf ("%c", &nextchar) ;
  - scanf ("%f", &length) ;
  - scanf (“%d %d”, &a, &b);

Comments in a C program

- Comments in C language are used to provide information about lines of code. It is widely used for documenting code.
- There are 2 types of comments in the C language.
  - Single Line Comments
  - Multi-Line Comments
- It is a good practice to begin a program with comment indicating the purpose of the program
- Comments should be enclosed within /* */.

Escape Sequence in C

- An escape sequence in C language is a sequence of characters that doesn't represent itself when used inside string literal or character.

- It is composed of two or more characters starting with backslash \.

- For example: \n represents new line.

List of Popular Escape Sequences in C

Escape Sequence	Meaning
\n	New Line
\t	Tab (Horizontal)
\v	Vertical Tab
\\	Backslash
\'	Single Quote
\"	Double Quote

What is main()?

- main() forms a crucial part of any c program.
  - main() is a function
    - A function is nothing but a container for a set of statements.
- All statements that belong to main() are enclosed within a pair of braces { } as shown below.

```
int main()
{
statement 1;
statement 2;
statement 3;
}
```

main() is also a function

```
#include <stdio.h>
main()
    {
        int   a, b, c;

        a = 10;
        b = 20;
        c = a + b;
        printf ("\n The sum of %d and %d is %d\n",
                    a,b,c);
    }
```

https://ideone.com/XQas3R

https://ideone.com/t0LFLT

Sample C program 1

Header file includes functions for input/output

```
#include <stdio.h>

main()
    {
        printf ("\n Our first look at a C program \n");
    }
```

Main function is executed when you run the program. (Later we will see how to pass its parameters)

Curly braces within which statements are executed one after another.

Statement for printing the sentence within double quotes (“..”). ‘\n’ denotes end of line.

Our first look at a C program

Sample C program 2

```
#include <stdio.h>
main()
    {
        int   a, b, c;
        a = 10;
        b = 20;
        c = a + b;
        printf ("\n The sum of %d and %d is %d\n",
                  a,b,c);
    }
```

Integers variables declared before their usage.

Control character for printing value of a in decimal digits.

The sum of 10 and 20 is 30

https://ideone.com/HHGLzd

Exercises

- Point out the errors, if any

```
#include<stdio.h>
int main()
{
int a, float b, int c;
a=25; b=3.24;  c=a+b*b-35;
}
```

Exercises

- Point out the errors, if any

```
#include<stdio.h>
int main()
{
int a, float b = 3.24;
printf("%d %f  %d", a, b + 1.5, 235);
}
```

Exercises

- Point out the errors, if any

```
int main()
{
int a, b, c;
scanf(“%d %d %d”, a, b, c);
}
```

Exercises

- Point out the errors, if any

```
int main()
{
int m1,m2,m3
printf(“Enter values of marks in 3 subjects)
scanf(“%d %d %d”,&m1,&m2,&m3)
printf(“You entered %d %d %d”,m1,m2,m3)
}
```

Exercises

- Attempt the following:

Ramesh's basic salary is input through keyboard. His dearness allowance is 40% of the basic salary, and house rent allowance is 20% of basic salary. Write a program to calculate his gross salary.

https://ideone.com/wIz4a9

Exercises

- Attempt the following:

If the marks obtained by a student in five different subjects are input through the keyboard, write a program to find out the aggregate marks and percentage marks obtained by the student. Assume that the maximum marks that can be obtained by a student in each subject is 100.

https://ideone.com/qqQAT9