The First Lecture –

Programming for Problem Solving (CS101)

PRESENTED BY: SATYA PRAKASH PATEL
EMAIL: SATYAPATEL.IND@GMAIL.COM

Syllabus

Module I **[9L]**

Introduction to Programming:

Introduction to components of a computer system (disks, memory, processor, where a program is stored and executed, operating system, compilers etc.)

Problem Solving: Steps to solve logical and numerical problems.

Representation of Algorithm: Flowchart/Pseudo code with examples. From algorithms to programs; source code, variables (with data types) variables and memory locations, Syntax and Logical Errors in compilation, object and executable code

Module II **[9L]**

Arithmetic expressions and precedence, Conditional Branching and Loops, Writing and evaluation of conditionals, Iterations, Loops.

Module III **[9L]**

Array, Character array, strings. Case studies to discuss the various Problems related to Basic science (Matrix addition, Matrix-matrix multiplication, Roots of an equation etc.), Sorting, Searching.

Module IV **[9L]**

Functions (including using built in libraries), Parameter passing in functions, call by value, call by reference. Passing arrays to functions, Recursion (Finding Factorial, Fibonacci series, Ackerman function etc.).

Module V **[9L]**

Structures, Defining structures and Array of Structures

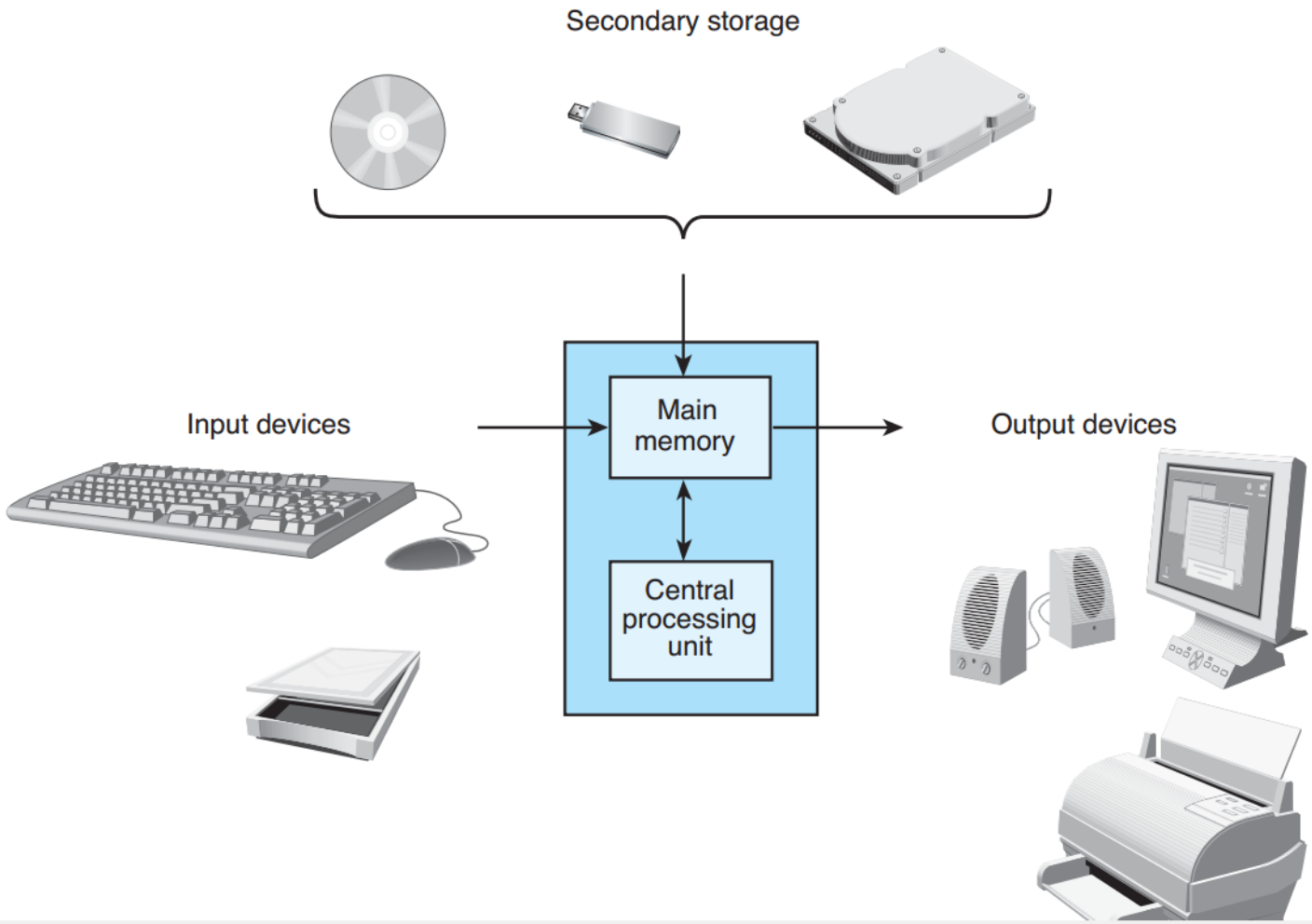
Pointers: Defining pointers, Use of Pointers in self-referential structures, File Handling

a

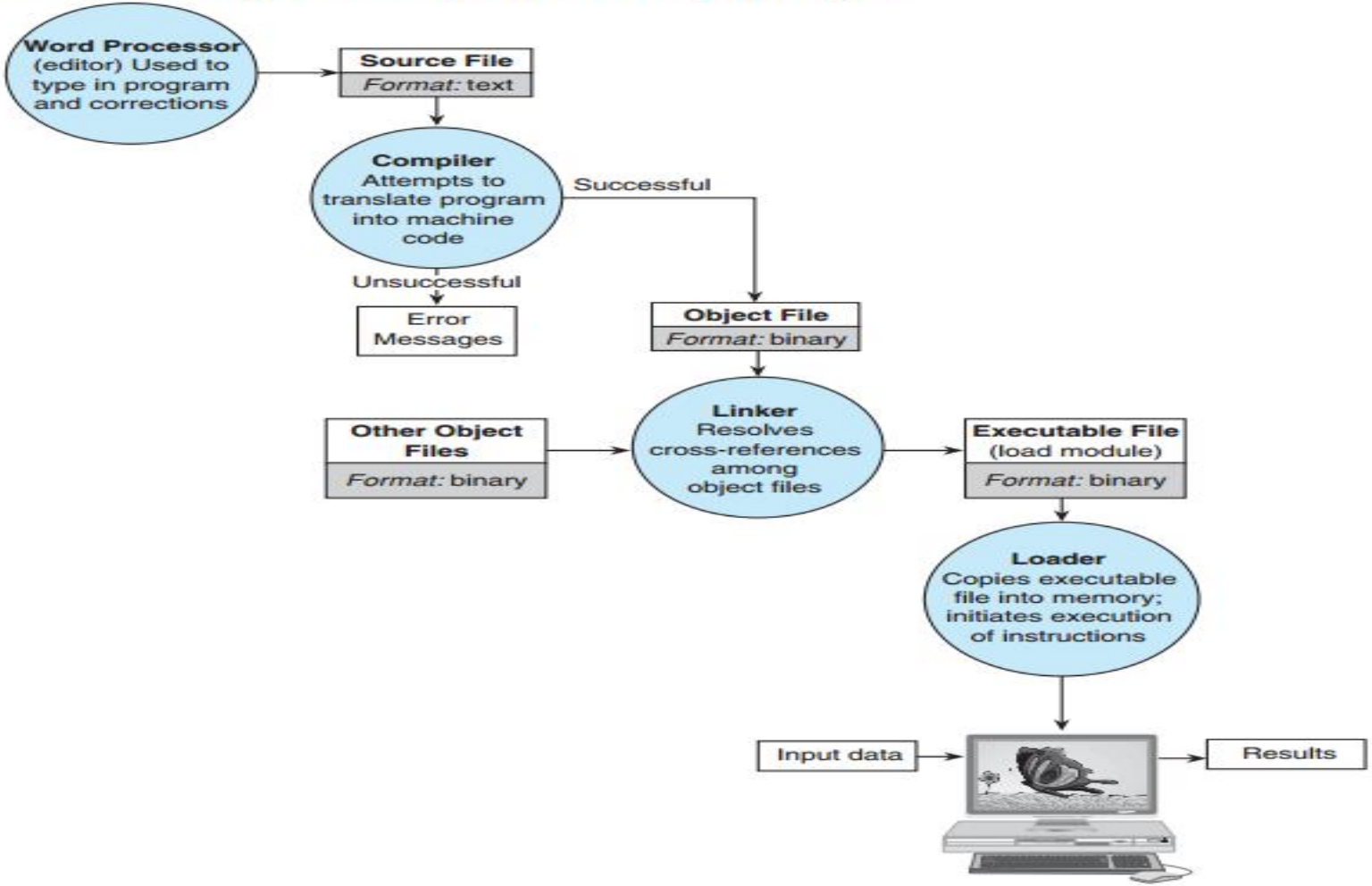
b

c

d

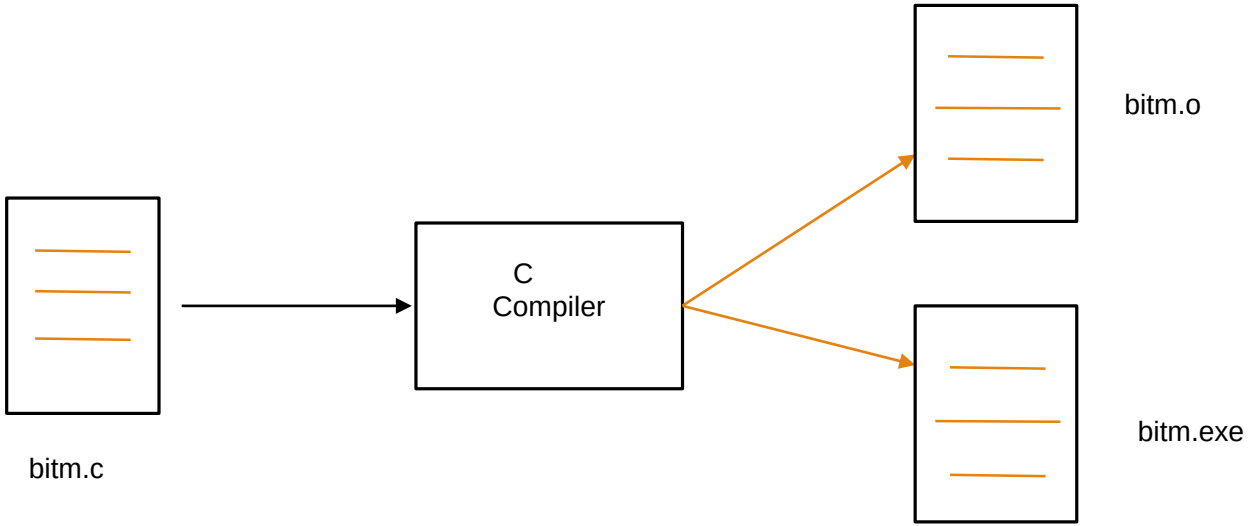

Secondary storage
Input devices
Main memory
Central processing unit
Output devices

Entering, Translating, and Running a High-Level Language Program



Behind the Scenes

- What happens when a C program is compiled ?



Programming on Linux

Writing programs

- Use any editor (graphical, console)
- Save file as <filename>.c

Compiling programs

- gcc <filename>.c gcc funnysingh.c –o funnysingh

Running programs

- ./a.out ./funnysingh
  (executable files need to have executable permissions. $chmod +x <executable>)

Compilation is not a single stage

Pre process : cpp (C Preprocessor) gcc –E

- Removes comments, includes #include files

Compile : gcc –c (GNU compiler)

- main step, compilation, change into machine code

Link : ld (GNU linker)

- link executables

gcc does all the above steps

C is a high-level language.

Writing a C code. {editors like gedit, vi}

Compiling a C code. {gcc –c test.c –o test}

Executing the object code. {./test}

Some more basics

Keywords

- char, static, if , while, return ..... Total= about 32

Data Types

- int , char, float ..... Some more later

Arithmetic Operators

- + (Plus), - (Minus), * (Multiplication), /(Division)

..... Some more later

Keywords

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

My first C program!

```
#include <stdio.h>
// program prints hello world
int main() {
    printf ("Hello world!");
    return 0;
}
```

Output: Hello world!

Example 1

```
#include <stdio.h>
// program prints a number of type int
int main() {
   int number = 4;
   printf (“Number is %d”, number);
   return 0;
}
```

Output: Number is 4

Example 2

```
#include <stdio.h>
// program reads and prints the same thing
int main() {
    int number ;
    printf (" Enter a Number: ");
    scanf ("%d", &number);
    printf ("Number is %d\n", number);
    return 0;
}
```

Output : Enter a number: 4

Number is 4

more and more

```
#include <stdio.h>

int main() {
  /* this program adds
  two numbers */
  int a = 4; //first number
  int b = 5; //second number
  int answer = 0; //result
  answer = a + b;
}
```

Note

Errors

Compilation

Compiler generally gives the line number at which the error is present.

Run time

C programs are sequential making the debugging easier.

Some more Data Types

Primary : int, float, char

- int (signed/unsigned)(2,4Bytes): used to store integers.
- char (signed/unsigned)(1Byte): used to store characters
- float, double(4,8Bytes): used to store a decimal number.

User Defined:

- typedef: used to rename a data type
  - typedef int *integer*; can use *integer* to declare an int.
- enum, struct, union

Rules for an Identifier

1. can only have alphanumeric characters(a-z , A-Z , 0-9) and underscore(_).
2. The first character of an identifier can only contain alphabet(a-z , A-Z) or underscore (_).
3. Identifiers are also case sensitive in C. For example **name** and **Name** are two different identifiers in C.
4. Keywords are not allowed to be used as Identifiers.
5. No special characters, such as semicolon, period, whitespaces, slash or comma are permitted to be used in or as Identifier.

Examples of Valid and Invalid Names

Valid Names		Invalid Names	
a	a1	$sum	/* $ is illegal */
student_name	stdntNm	2names	/* Starts with 2 */
_aSystemName	_anthrSysNm	stdnt Nmbr	/* no spaces */
TRUE	FALSE	int	/* reserved word */

Some more Arithmetic Operators

Prefix Increment : ++a

- example:
  - int a=5;
  - b=++a; // value of b=6; a=6;

Postfix Increment: a++

- example
  - int a=5;
  - b=a++; //value of b=5; a=6;

Contd…

Modulus (remainder): %

- example:
  - 12%5 = 2;

Assignment by addition: +=

- example:
  - int a=4;
  - a+=1; //(means a=a+1) value of *a* becomes 5

Can use -, /, *, % also

Contd…

Comparision Operators: <, > , <=, >= , !=, ==, !, &&, || .

- example:
  - int a=4, b=5;
  - a<b returns a true(non zero number) value.

Bitwise Operators: <<, >>, ~, &, | ,^ .

- example
  - int a=8;
  - a= a>>1; // value of a becomes 4

Operator Precedence

Meaning of a + b * c ?

is it a+(b*c) or (a+b)*c ?

All operators have precedence over each other

*, / have more precedence over +, - .

- If both *, / are used, associativity comes into picture. (more on this later)
- example :
  - 5+4*3 = 5+12= 17.

Precedence Table

Highest on top
++ -- (Postfix)
++ -- (Prefix)
* / %
+ -
<< >>
< >
&
|
&&
||

Input / Output

- printf (); //used to print to console(screen)
- scanf (); //used to take an input from console(user).
  - example: printf("%c", 'a'); scanf("%d", &a);
  - More format specifiers

%c	The character format specifier.
%d	The integer format specifier.
%i	The integer format specifier (same as %d).
%f	The floating-point format specifier.
%o	The unsigned octal format specifier.
%s	The string format specifier.
%u	The unsigned integer format specifier.
%x	The unsigned hexadecimal format specifier.
%%	Outputs a percent sign.

Some more geek stuff

& in scanf.

- It is used to access the address of the variable used.
- example:
  - scanf(%d,&a);
  - we are reading into the address of a.

Data Hierarchy.

- example:
  - int value can be assigned to float not vice-versa.
  - Type casting.

Home Work

Meaning of

- Syntax
- Semantics of a programming language

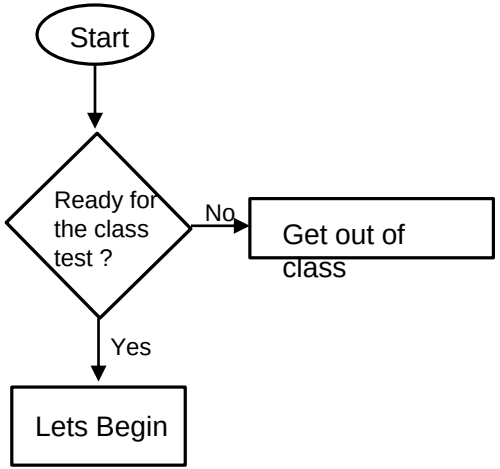
Find the Output:

- value=value++ + value++;
- Value=++value + ++value;
- value=value++ + ++value;

How a problem is solved ?

Step 1:- Draw a flowchart.
Step 2:- Write an algorithm according to flowchart.
Step 3:- Choose any programming language to formulate the algorithm into code.

Flowchart



Algorithm

1. Start
2. Ready for the class test ?
3. If Yes, goto 5
4. If No, goto 6
5. Lets Begin
6. Get out of class
7. Exit

Code

```
#include<stdio.h>
void main()
{
class_test=0;
if(class_test==1)
printf(“Lets Begin!”);
else
printf(“Get out of class”);
```

My First C Program

- **What is your roll number ?**

```
#include <stdio.h>
void main()
{
  int roll_no;
  printf("\n What is your roll number:");
  scanf("%d", &roll_no);
  printf(" \n Your roll number is:%d",roll_no);

}
```

Good programming practices

Indentation

```
#include <stdio.h>
int main() {
   printf("Hello World!\n");
   return 0;
}
```

```
#include <stdio.h>
int main() {
printf("Hello World!\n");
return 0;
}
```

Good programming practices contd..

Variables names

- Not too short, not too long
- Always start variable names with small letters
- On work break
  - Capitalize: myVariable, OR
  - Separate: my_variable

Good programming practices contd...

Put comments

```
#include <stdio.h>
int main() {
  /* this program adds
  two numbers */
  int a = 4; //first number
  int b = 5; //second number
  int res = 0; //result
  res = a + b;
}
```

Good programming practices

Your code may be used by somebody else

The code may be long

Should be easy to understand for you and for others

Saves lot of errors and makes debugging easier

Speeds up program development

Queries and Feedback