Introduction to Programming Methodology and Problem Solving Strategies

Lecture 1 – ICT1132

Piyumi Wijerathna
Department of ICT
Faculty of Technology

Objectives

- Learn about problem solving skills.
- Explore the algorithmic approach for problem solving.
- Become aware of problem solving process.
- Understanding of first C program.

Problem Solving

- Problem solving techniques:
  - Analyze the problem
  - Outline the problem requirements
  - Design steps (algorithm) to solve the problem
  - Implement the solution
  - Test and debug
- Algorithm:
  - Step-by-step problem-solving process
  - Solution achieved in finite amount of time

Programming

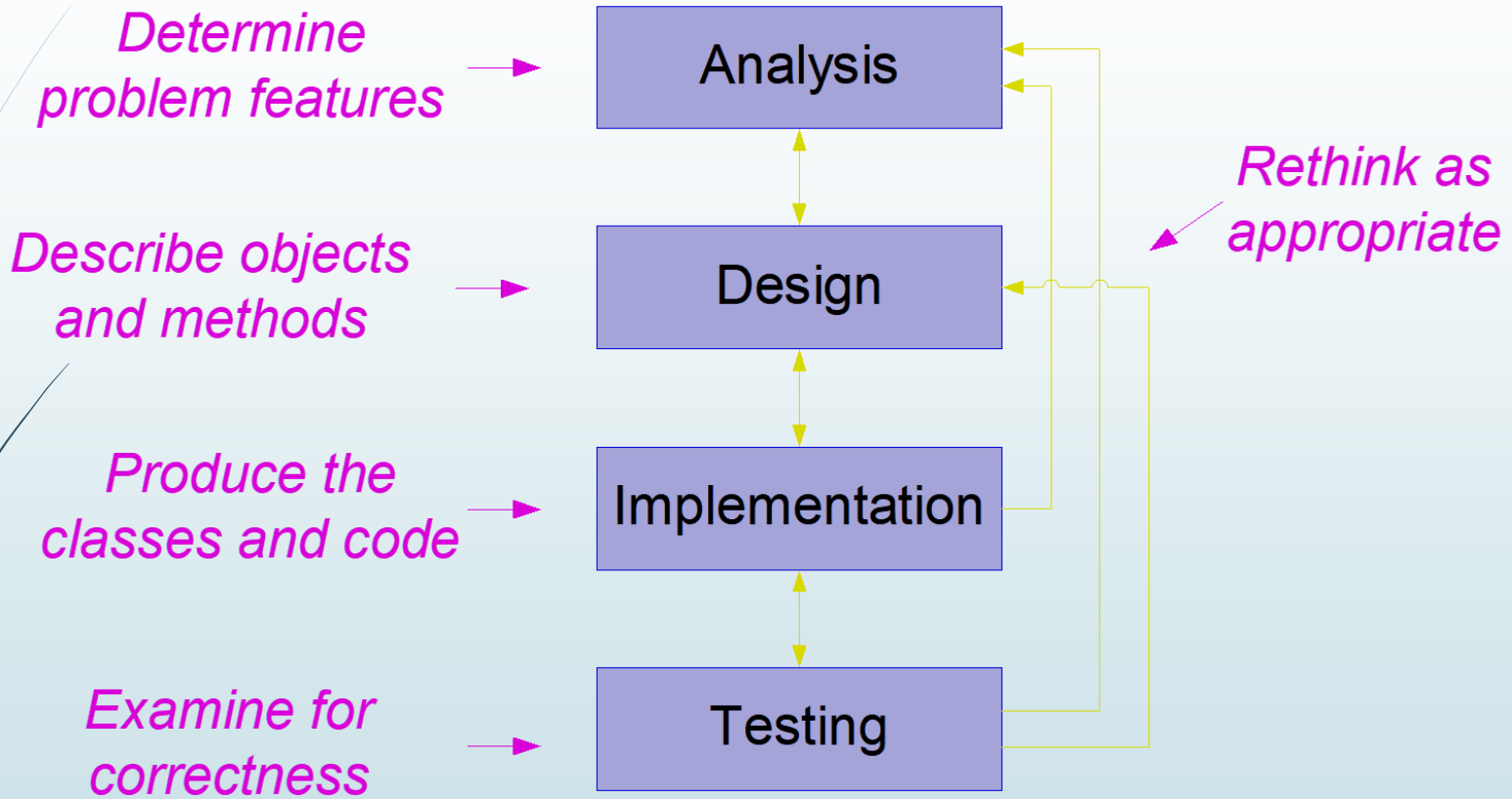
- It is a process of problem solving.
- Writing sequences of instructions in order to organize the work of the computer to perform something.
- These sequences of instructions are called "computer programs" or "scripts".
- Programming involves describing what you want the computer to do by a sequence of steps, by algorithms.
- Programmers are the people who create these instructions, which control computers.

Program Design

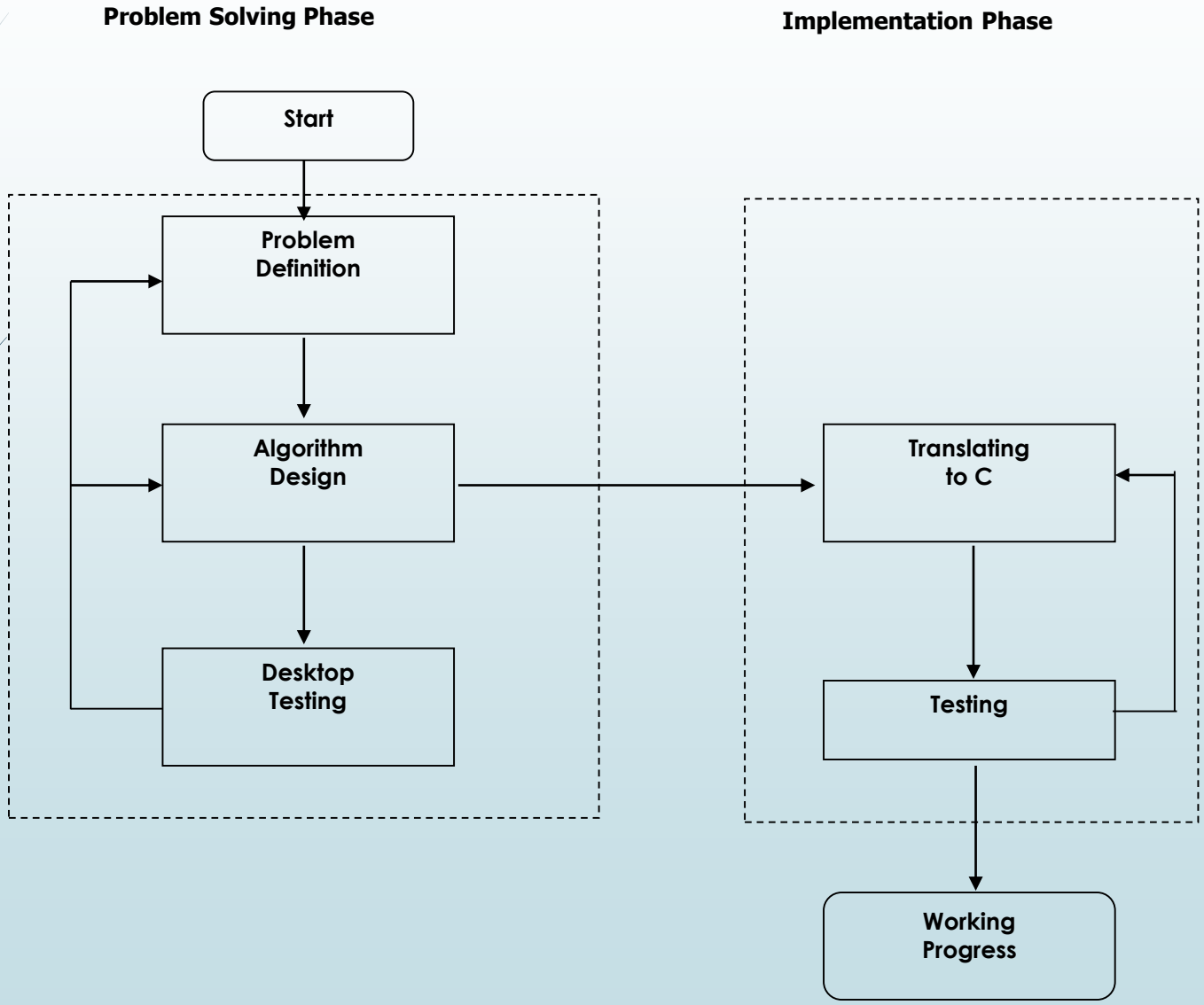
- Top-Down Design to writing a program is similar as making an outline to writing a report.
- Importance
  - Provides organization and structure to ideas.
  - As size gets larger, its importance increases.
  - Goes from the general ideas to the specific details.

Design and Development

1. Analyze the problem
2. Develop a solution
3. Code the solution
4. Test and Debug


Determine problem features
Analysis
Describe objects and methods
Design
Produce the classes and code
Implementation
Examine for correctness
Testing
Rethink as appropriate

Writing Programs



1. Analyze the Problem

- **Thoroughly understand the problem.**
- Determine the information, the computer has to produce, i.e. the output.
- Determine the information required to produce the output, i.e. the input data.
- Determine if any special cases exist, and how they are to be handled.

Understand Problem Requirements

- Does program require user interaction?
- Does program manipulate data?
- What is the output?
- The requirements are defined by experts, who are familiar with the problems in a certain field.
  - requirements for the solution are usually defined in the form of documentation, written in English or any other language.
- No programming is done at this stage.

2. Develop the Solution

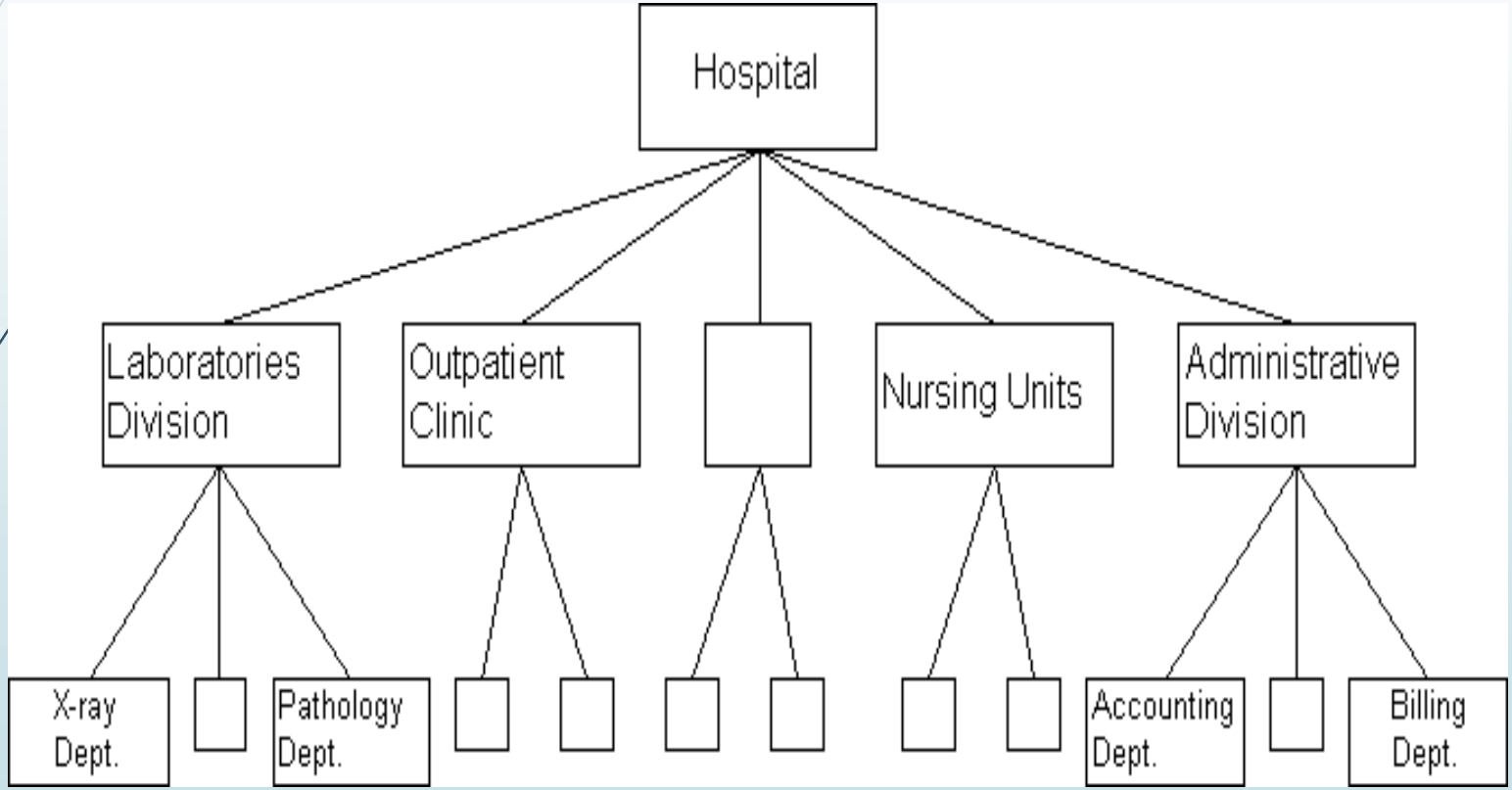
- Development of an algorithm to solve a problem which can transform the input to the output.
- If the problem is complex, divide the problem into subtasks, where the completion of all the subtasks in some specified order, will produce the desired output.

Top-Down Design

Top-down design is a design technique in which,

- Divide the work that a program will perform (Problem Solution) into more manageable subtasks (Modules).
- First understand the complete solution for the problem.
- Then divide the solution into detailed sub tasks.
- Divide each sub modules in the same manner as required.

Top down design of a Hospital



Design Components

- Data Structures
  - Description of data, its name, type, and range of values.
- Algorithms
  - An algorithm is a precise, unambiguous, step-by-step description of how to do a task in a finite amount of time.
  - Independent of any particular programming language.

3. Code the Solution

- Write the algorithm in pseudo code.
- Convert your pseudo code to a source code using a programing language.
- Look up the syntax of any instruction statement you are unsure of as you write them.
- Compile the program, and fix any syntax errors.

4. Test and Correct the Program

- Determine a set of test data, and calculate the expected answer.
- Run the program, and compare the computer's answer with the expected answer.
- If they do not match, examine the program; determine where the error is; and fix the error.
- **Do not make random changes hoping the error will simply disappear by luck.**

Applying the Design Process to an Example

- In an experiment, the height of 5 people were recorded and the average height was computed. Due to unconcern the height of two people have been lost. The lab assistant distinctly remembers that the two missing people were almost the same height.
- You are required to develop an algorithm that inputs the heights of the known people and the calculated average. Determine the missing height.

Step 1: Understand the problem

The height of 5 people were recorded and the average height was computed.

Due to negligence the height of two people have been lost?????

The lab assistant distinctly remembers that the two missing people were almost the same height.

Requirement is????

- Develop an algorithm that inputs the heights of the known people and the calculated average.
- Determine the missing height???

Analyze the problem

- What are the Inputs ?
- What are the Outputs ?
- Calculations ?
- Background

Height of 3 people,
Average height

Missing height

Missing height ?

Additional information:

Missing heights are same

Step 2: Design

- **Algorithmic Design**
  - Input heights of 3 people
  - Input average height
  - Find missing height
  - Print missing height

Step 2: Design

- Find Missing height
  - calculate sum of 3 people
  - Missing_height = (avg_height *5 – sum)/2

Step 2:Design

- Complete Algorithm
  - Input heights of 3 people
  - Input avg height
  - calculate sum of 3 people
  - missing height = (avg height *5 – sum)/2
  - Print missing height

Step 3: Code Solution

Algorithm

Input heights of 3 people

Input average height

calculate sum of 3 people

missing height = (average height *5 – sum)/2

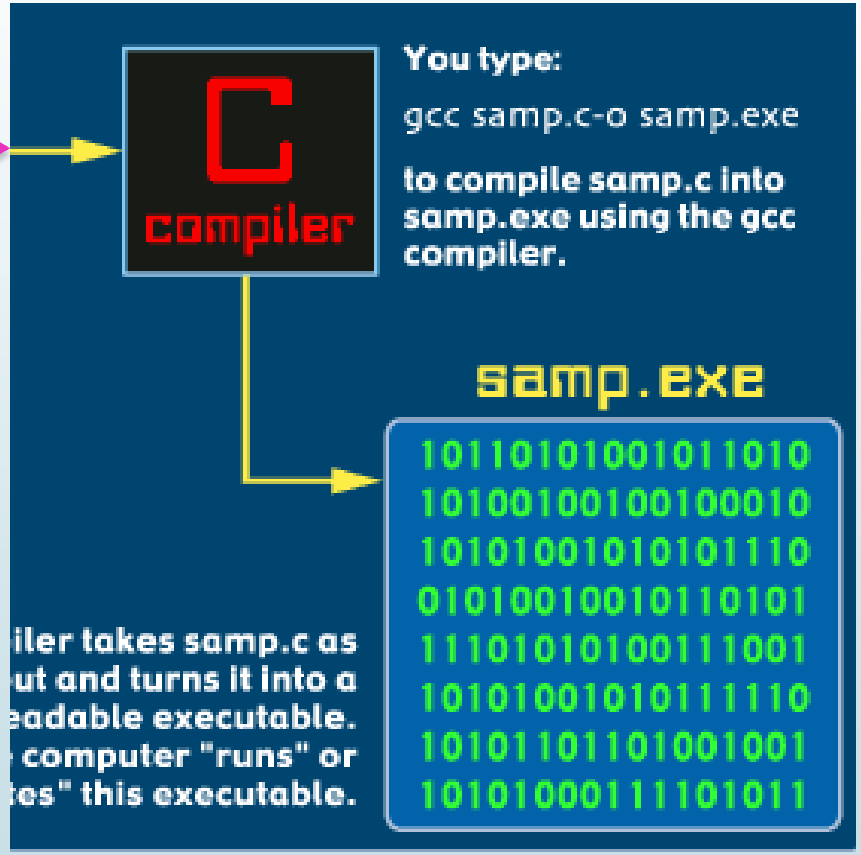
Print missing height1

C program

```
#include<stdio.h>
int main()
{
 float height1, height2, height3;
 float sum average,missing_height;

//Input heights of know persons
 printf("Input height of person1?");
 scanf("%d",height1);
 printf("Input height of person2?");
 scanf("%d",height2);
 printf("Input height of person3?");
 scanf("%d",height3);

//Input average height
 printf("Input average height  of people?");
 scaff("%d",average);

//Calculating sum and avearge
 sum= height1+height2+height3;

//Calculating missing height
 missing_height= (average*5-sum)/2;

//Printing Missing height of missed people
  printf("Missing height is %d", missing_height);

return 0;
}
```

Step 4: Test program

```
#include<stdio.h>
int main()
{
 float height1, height2, height3;
 float sum average,missing_height;

//Input heights of know persons
 printf("Input height of person1?");
 scaff("%d",height1);
 printf("Input height of person2?");
 scaff("%d",height2);
 printf("Input height of person3?");
 scaff("%d",height3);

//Input average height
 printf("Input average height  of people?");
 scaff("%d",average);

//Calculating sum and avearge
 sum= height1+height2+height3;

//Calculating missing height
 missing_height= (average*5-sum)/2;

//Printing Missing height of missed people
  printf("Missing height is %d", missing_height);

return 0;
}
```



Programming Language Translation

- Assemblers
- Interpreters
- Compilers

- Computers can only understand machine code.
- A translator will convert the source code into machine code.
- There are several types of translator programs.

Program Language Translation

Assemblers

- Translate statement from assembly language to machine language.
- One low-level language statement is usually translated into one machine code instruction.
- Generally use table look-up techniques.

Interpreters

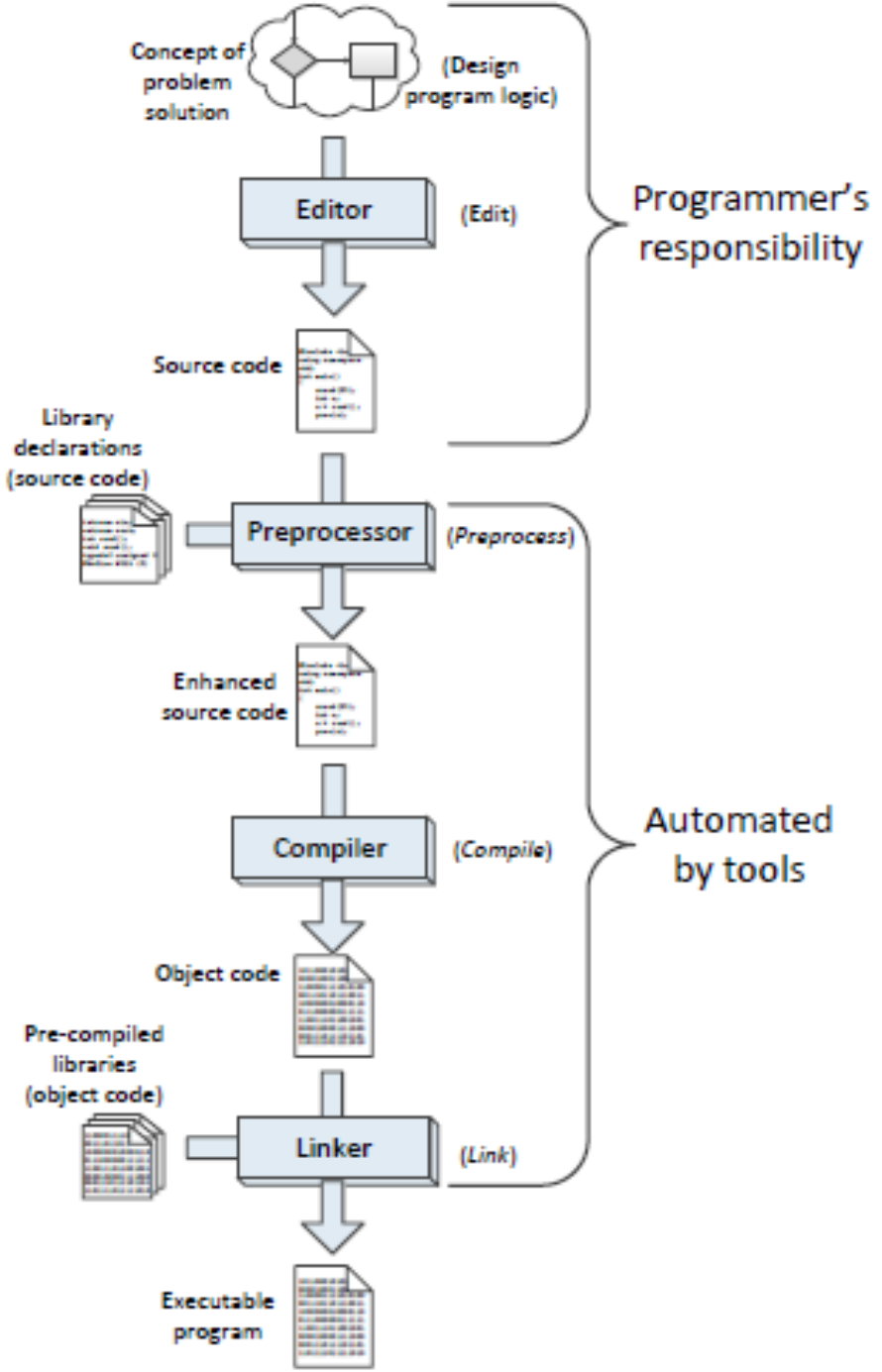
- Translate and execute each high-level statement, one at a time.
- Error message produced immediately (program stops at that point).

Program Language Translation

Compilers

- Translate the entire high level language program, called the source code, to machine language and store the result.
- Machine language code produced by compilers is called the object code.
- An error report is often produced after the full program has been translated.

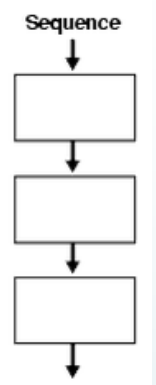
C Program Compilation Process



Control Structures

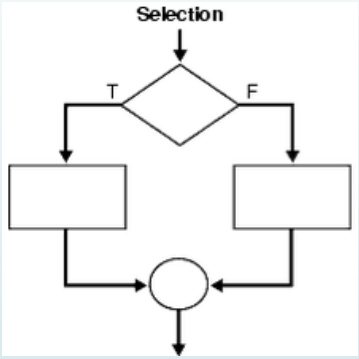
Sequence

- Execute sequence of statements in order given.
- This is the default control structure.



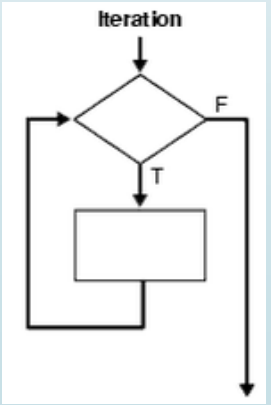
Selection

- Select between alternative paths.



Iteration

- Repeat a statement or collection of statements over and over again.



Invocation

- Execute a complex task, which is written separately.

Introduction to C programming Language

Origin of C language

Ken Thompson (left) with Dennis Ritchie (right, the inventor of the C programming language)

- The C programming language was devised in the early 1970s by Dennis M. Ritchie and K. Thompson.

- 1960s Ritchie worked in Bell Labs (AT&T), on a project called Multics (develop an operating system for a large computer)-failed.
- UNIX system was developed - fully based on Assembly codes.
- The language B was developed in 1969-70 by Ken Thompson.
- The language B was used for further development of the UNIX system, drawback – no data-types.

- 1971-73 Dennis M. Ritchie turned the B language into the C language, keeping most of the language B syntax while adding data-types and many other changes.
- Unix kernel was rewritten in C in 1973 on a DEC PDP-11.
- The programming language C was written down, by B.Kernighan and D.Ritchie, in a book called "The C Programming Language"; it is now known as **'K&R C'**).
- *1978-1985:* C compilers became available on different machine architectures.
- *1983-1990:* ANSI established committee **X3J11** with a "*goal of producing a C standard*"; (standard **ISO/IEC 9899-1990**).

C language keywords

- C is very "*compact*" language (**32 keywords** in ANSI C = **27 keywords** in K&R C + **5 keywords** which were added by ANSI committees)

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

All keywords are lowercase!

C - Program Structure

A C program basically has the following form:

- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

First C Program

```
//File: welcome.c
//A program to print a welcoming
 message

#include <stdio.h>     //preprocessor
int main()             // main function

{
 printf("Welcome to
 Ruhuna\n");//statement
  return 0;
} //close main function
```

First C Program

Note the following:

- Comments of the program.
- The include file is **`stdio.h`**
- The **`main()`** function has a return value.
- The use of **`printf()`** for outputting to the monitor.
- The use of **`\n`** for new line.
- The use of **`return 0`** for exitting the program without errors.

// File: welcome.c
// A program to print a welcoming message

- Comments provide information to the people who read the program.

- Comments are removed by the preprocessor, therefore the compiler ignores them.

- In C there are two types of comments
  - Line comments
  - Delimited comments

More on Comments

- Line comments begin with // and continue for the rest of the line.
- Delimited comments begin with /* and end with */

→ **Inline comments**

```
// This is an inline comment
```

→ **Block comments**

```
/* This is a block comment.
   It can span multiple lines. */
```

#include

- ***#include*** is a preprocessor directive. Lines beginning with # are processed by the preprocessor before the program is compiled.
- This specific line tells the preprocessor to include in the program the contents of the input/output stream header file ***<stdio.h>***
- This file must be included for any program that outputs data to the screen or inputs data from the keyboard using C style stream input/output.

int main( )

- *int main()* is a part of every C program.
- C programs can contain one or more functions, but only one main function.
- The parentheses after main indicate that it is a program building block called a function and it is similar to other functions.
- The only difference is that the main function is "called" by the operating system when the user runs the program.
- The keyword *int* indicates that main function returns an integer value.
- The **{** indicates the begin of the main function.

printf( )

- **printf("Welcome to Ruhuna\n");**
- It instructs the computer to print on the screen the string of characters contained between the quotation marks.
- Here **\n** is the new line control character, this moves the cursor to the next line.
- Every statement in C must end with a semicolon (;).

return 0;

- The return statement is used to return a value from a function.
- The value 0 indicates that the program has terminated successfully.
- The right brace }, indicates the end of the main function.

Free Format

- A language has free format if statements can appear anywhere on one or several lines.
- Blank lines are ignored. Extra spaces are ignored.
- Multiple statements can be placed on the same line.
- Statements can be broken over more than one line anywhere other than in the middle of a string literal, operator, or identifier.

Program Style

- Use blank lines to separate parts of the program.
- Use indentation to align curly braces and match the opening and closing braces by placing them under each other.
- Use blank spaces to separate the parts of a statement to make it easier to read.


Any
Question

THANK YOU... !