ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

Introduction to Visual and Interactive Programming

CT803-4-0-OIVIP

Week 1

Module Overview and History of Programming

Lecturer Information

Lecturer Name: Mohd Najmuddin Suki

Email: najmuddin.suki@apu.edu.my

Consultation Hours: By appointment

Pre-Requisites For This Module

- N/A

Outcomes Based Education

OBE is education based on producing particular educational outcomes that:

Focus on what students can do after they are taught.

Expect all learners / students to successfully achieve (sometimes minimum) level of knowledge and abilities.

It’s NOT what We want to teach.

It’s WHAT You should learn.

Aims of this Module

Computational thinking is a skill to **solve a problem logically** via computer programming means. For this module, the aim is to apply **visual and interactive** programming elements. Students will learn the essential skills required in designing and implementing software solutions regardless of platform, language, or application domain. This module will cover the following four elements: decomposition, pattern recognition, algorithm and abstraction.

Module Learning Outcomes

CLO	Learning Outcomes	Assessment
1	Describe the principles of Computational Thinking (C1, PLO1)	Final Exam
2	Apply the elements of computational thinking to solve a problem (C3, PLO2)	Project - Documentation
3	Use the visual interactive programming tools to develop an application. (A1, PLO6)	Project - Implementation

Mapping of CLO with PLO

	PLO 1	PLO 2	PLO 3	PLO 4	PLO 5	PLO 6	PLO 7	PLO 8	PLO 9	PLO 10	PLO 11	PLO12
CLO1	✓											
CLO2		✓										
CLO3						✓						

The learning domains are:
PLO1: Knowledge and Understanding
PLO2: Cognitive Skills
PLO6: Digital Skills

Student Learning Time

- Module Credit Value: 4
- Total Learning Hours: 160 per semester

Lecture	28 hours
Tutorial	28 hours
Practical	-
Others	-
Independent Learning Time	101 hours
Assessment	3 hours
Total Learning Hours	160 hours per semester

Module Content Outline

Week	Topic
1	Module Overview
2	Introduction to Computers and Programming
3-4	Introduction to Computational Thinking and Problem Solving
5	Data and Variables
6	Boolean expressions and operators
7-8	Control Structure: Selection and Loop
9	List
10	Procedures & Functions
11-12	Object Oriented Programming with Sprites and Procedure
13	Assignment Submission and Presentation
14	Exam Revision and Presentation

Assessment Summary (refer to module handbook and module descriptor)

Form of Assessment	Assessment Methods	Hand Out Date	Hand In Date	%
Continuous Assessment	Final Exam			50%
Final Assessment	Project - Documentation			15%
	Project - Implementation			35%

Assessment requirement: Include any specific requirement to pass the module (refer to module handbook for the information), such as:

- To pass the module, you must attempt every element of assessment and achieve at least 50% in the module overall.

Other Expectations

The art and science of asking questions is the source of all knowledge.
- Thomas Berger

- All of us learn through asking and being curious within a conversation
  - Do not hesitate to ask!
  - If something is not clear, stop me and ask.
  - During exercises / tutorial (you can also ask others).
- Failure
  - Coding is all about trial and error.
  - Don't be afraid of it.
  - Error messages aren't scary, they are useful.

Other Expectations – Group Assignment

- What is to be expected from everyone:
  - Clear Communication
  - Mutual Respect
  - Sense of Responsibility

Achievement Requirements:

Undergraduate (Diploma, Foundation, Degree)

Marks	Alphabetical Grade	Grading Point	Classification
80-100	A+	4.0	Distinction
75-79	A	3.7	
70-74	B+	3.3	Credit
65-69	B	3.0	
60-64	C+	2.7	Pass
55-59	C	2.3	
50-54	C-	2.0	
40-49	D	1.7	Fail (marginal)
30-39	F+	1.3	Fail
20-29	F	1.0	Fail
0-19	F-	0	Fail

Reference Materials

Course Materials available in Moodle

- Module handbook
- Module descriptor
- Lecture slides
- Tutorial/Lab materials
- Sample incourse questions & answers
- Sample exam questions & answers

Essential and Further Readings

- Mailund, T. (2021). Introduction to Computational Thinking: Problem Solving, Algorithms, Data Structures, and More (1st ed.). Apress, ISBN-10: 1484270762 and ISBN-13: 978-1484270769.
- Sweigart, A. (2021). Scratch 3 programming playground: Learn to program by making Cool Games (2nd ed.). No Starch Press, ISBN-10 : 1718500211 and 978-1718500211
- Tin Yu, C., & TomorrowSKILLS, H. (2020). Introduction to Block Based Programming: with Snap! (STEM Programming and Coding). HobbyPRESS TomorrowSKILLS.
- Mc Manus, S. (2019). Scratch Programming in easy steps (2nd ed.). In Easy Steps Limited, ISBN-10 : 1840788593 and ISBN-13 : 978-1840788594

*Further readings will be assigned from time to time.

Your Valuable Feedback

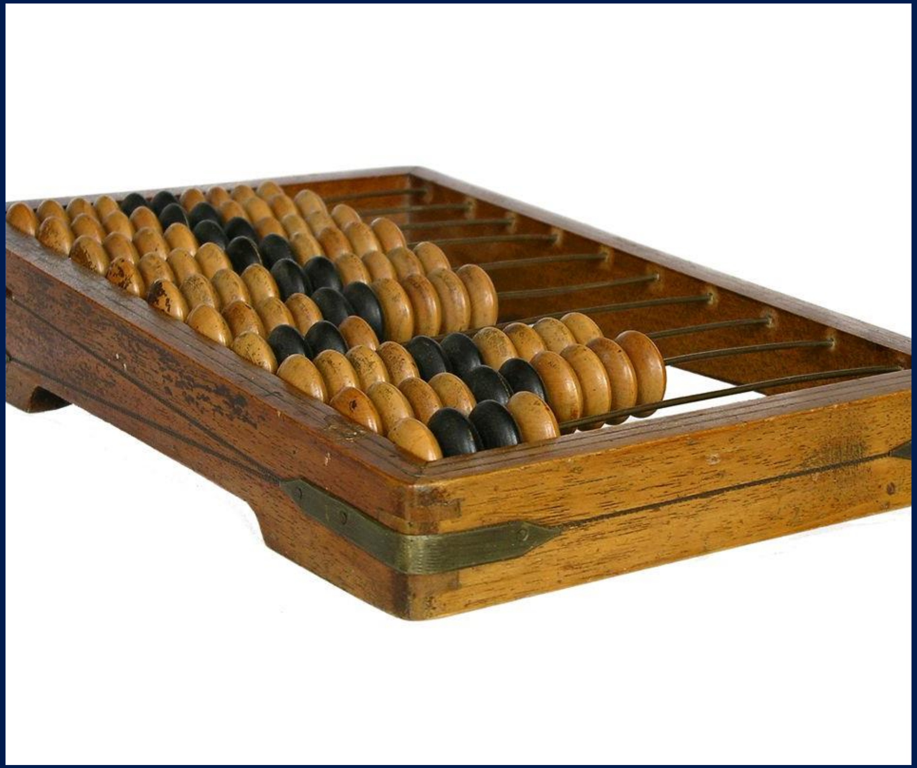
You are welcome to discuss your views on this module at any point of time.

Through your module lecturer or,

Do fill in anonymous evaluation questionnaires in the student feedback form. There are two points - mid and end of the teaching semester.

Early Computing Devices

- **Abacus** – early calculation tool
- **Charles Babbage** – father of the computer
    - – Analytical Engine
    - – Faced financial difficulties

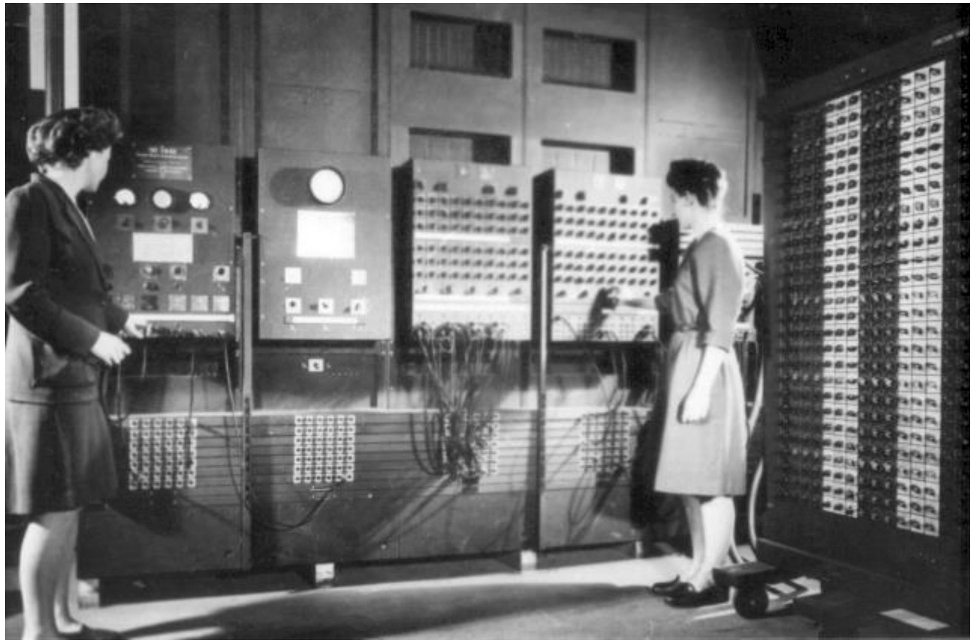


Birth of Programming

- **Ada Lovelace** – mathematician
  - “First programmer”
  - World’s first machine algorithm

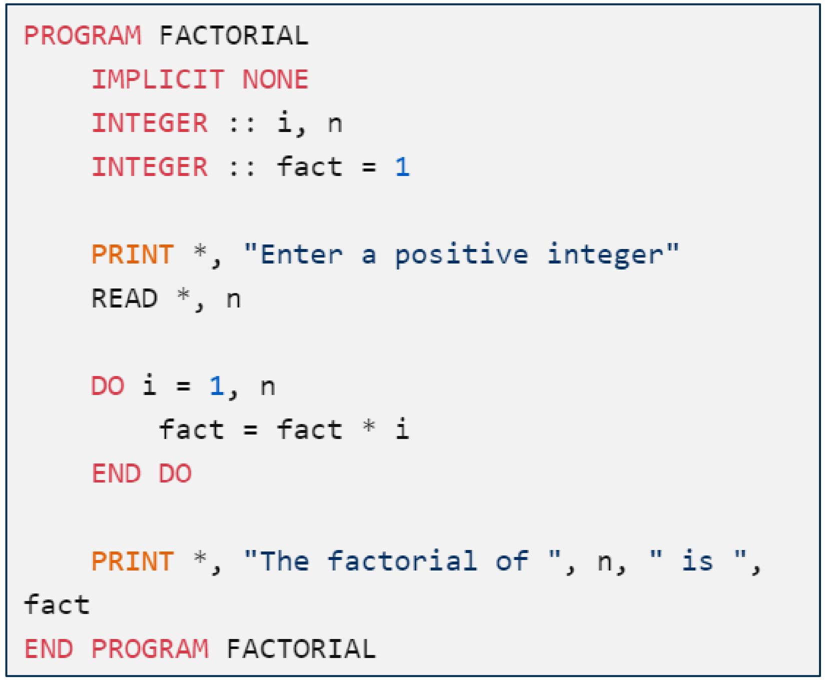
The Advent of Modern Computers

- **20th century** – birth of programmable computers
  - Development of electrical circuits, transistors, silicon chips
  - ENIAC
  - UNIVAC



Birth of High-Level Programming Languages

- Evolution of computers and tasks they perform
- **FORTRAN** – high-level programming language
  - Developed by IBM in the 1950s
  - Revolutionized human-computer interaction



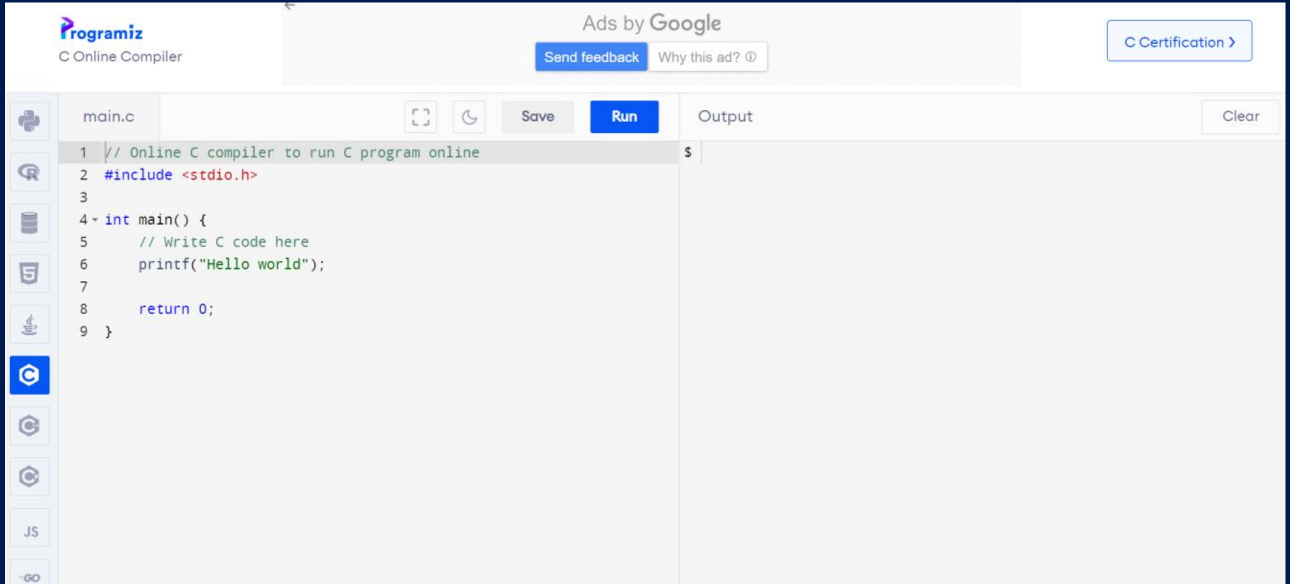
```
PROGRAM FACTORIAL
    IMPLICIT NONE
    INTEGER :: i, n
    INTEGER :: fact = 1

    PRINT *, "Enter a positive integer"
    READ *, n

    DO i = 1, n
        fact = fact * i
    END DO

    PRINT *, "The factorial of ", n, " is ",
fact
END PROGRAM FACTORIAL
```

Personal Computer Revolution

- Invention of the microprocessor
  - Computers become smaller, faster, cheaper
- Computers used by a person in homes/offices
  - Personal computer
  - Apple II, IBM PC, Windows-based computers

The Internet and Beyond

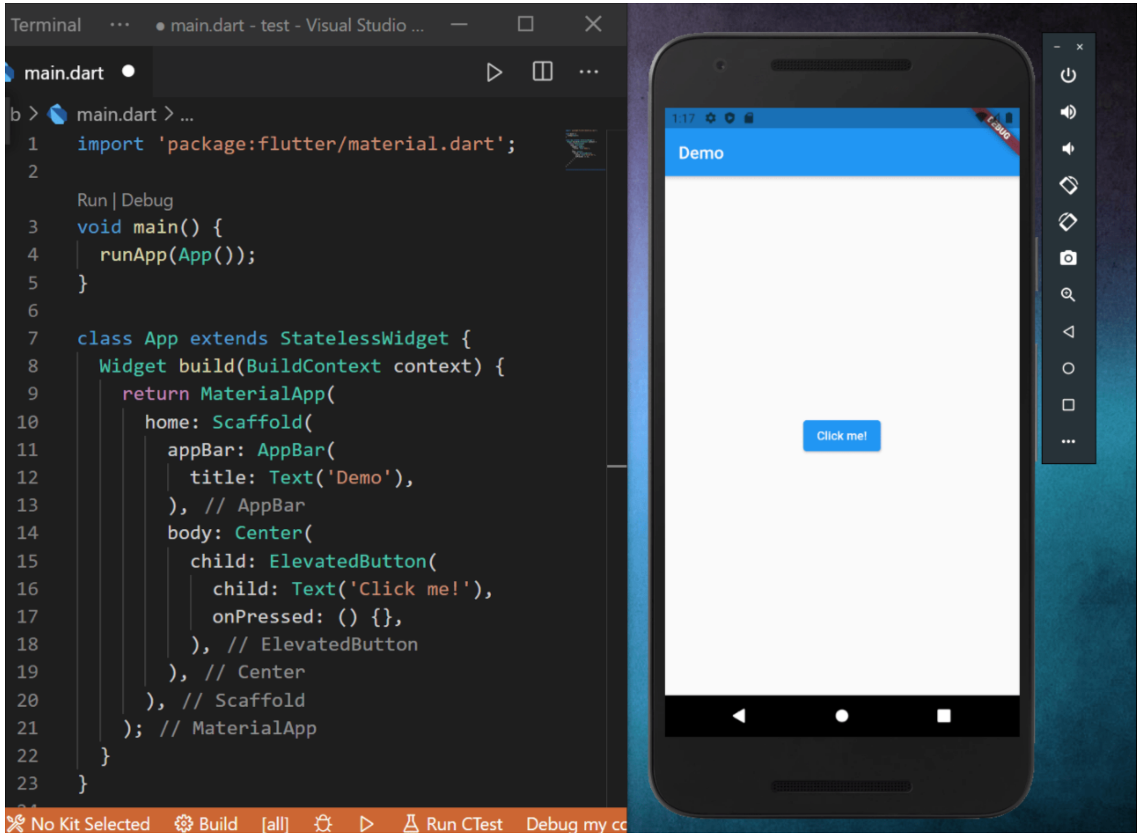
- Development of multiple languages
  - HTML, Javascript
- Programming becomes more accessible
  - High level languages such as Python
  - Web-based programming environment



```
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    // Write C code here
    printf("Hello world");

    return 0;
}
```

```
$
```

Computers and Programming



```
import 'package:flutter/material.dart';

void main() {
  runApp(App());
}

class App extends StatelessWidget {
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Demo'),
        ), // AppBar
        body: Center(
          child: ElevatedButton(
            child: Text('Click me!'),
            onPressed: () {},
          ), // ElevatedButton
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```