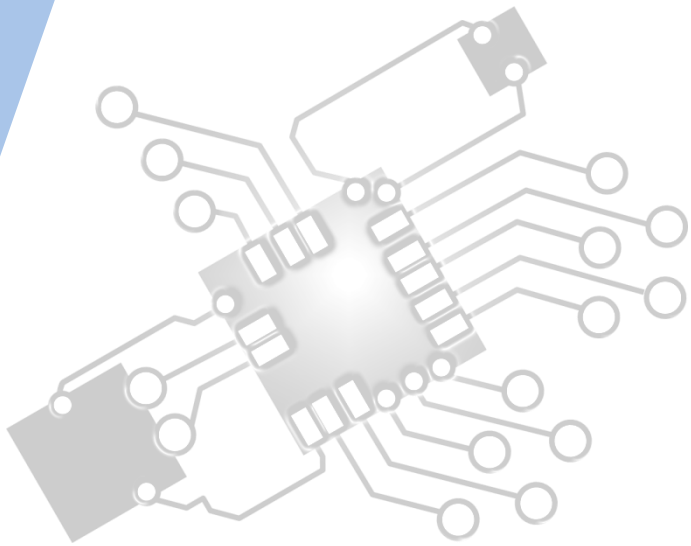
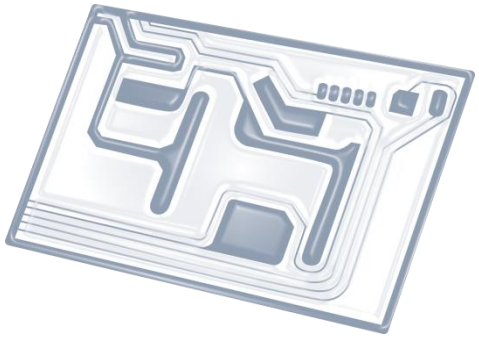
Computational thinking, problem-solving and programming: Introduction to programming

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL & SL 4.3 Overview

Nature of programming languages

4.3.1 State the fundamental operations of a computer

4.3.2 Distinguish between fundamental and compound operations of a computer

4.3.3 Explain the essential features of a computer language

4.3.4 Explain the need for higher level languages

4.3.5 Outline the need for a translation process from a higher level language to machine executable code

Use of programming languages

4.3.6 Define the terms: variable, constant, operator, object

4.3.7 Define the operators =, ., <, <=, >, >=, mod, div

4.3.8 Analyse the use of variables, constants and operators in algorithms

4.3.9 Construct algorithms using loops, branching

4.3.10 Describe the characteristics and applications of a collection

4.3.11 Construct algorithms using the access methods of a collection

4.3.12 Discuss the need for sub-programmes and collections within programmed solutions

4.3.13 Construct algorithms using predefined sub-programmes, one-dimensional arrays and/or collections


1: System design

2: Computer Organisation



3: Networks

4: Computational thinking

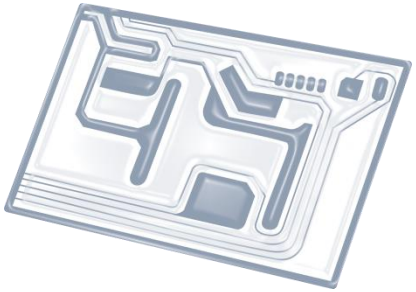

5: Abstract data structures

6: Resource management



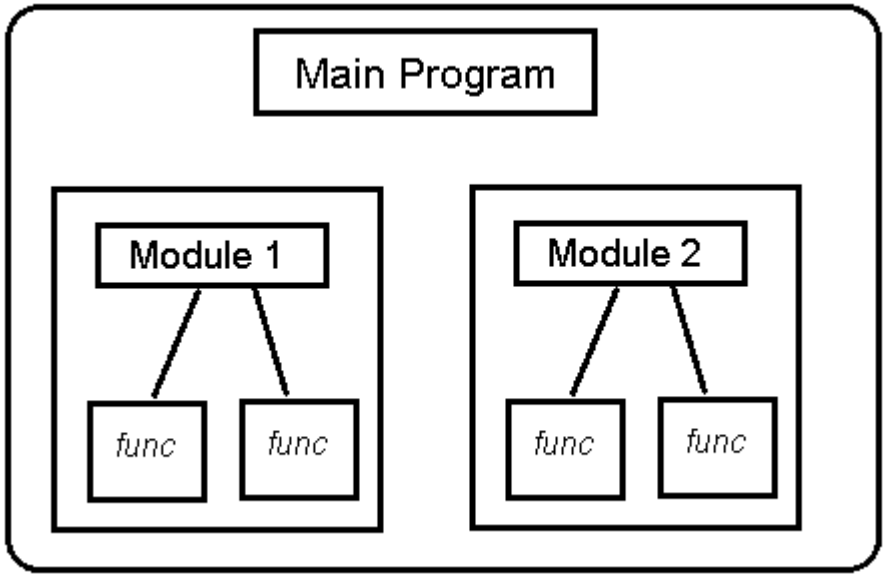
7: Control

D: OOP

Topic 4.3.12

Discuss the **need** for **sub-programmes** and **collections** within programmed solutions



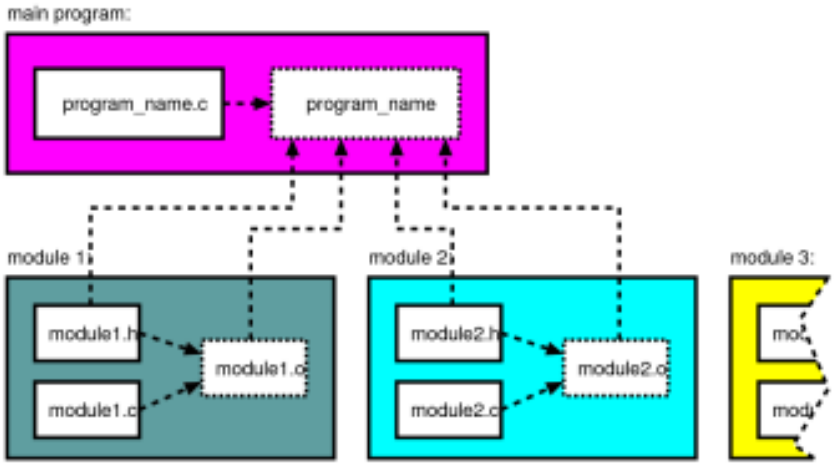
Advantages to modular design

- Modular programming is an important and beneficial approach to programming problems. They make program development easier, and they can also help with future development projects.

- **Key benefits/advantages**:
  - Usefulness of **reusable code**
  - **Eases program organization**, both for the individual programmer, team members
  - **Makes future maintenance easier** – you only have fix/update a module, not the whole program

Manageable tasks

- Breaking down a programming project into modules makes it more manageable.
- These individual modules are easier to design, implement and test.
- Then you can use these modules to construct the overall program.

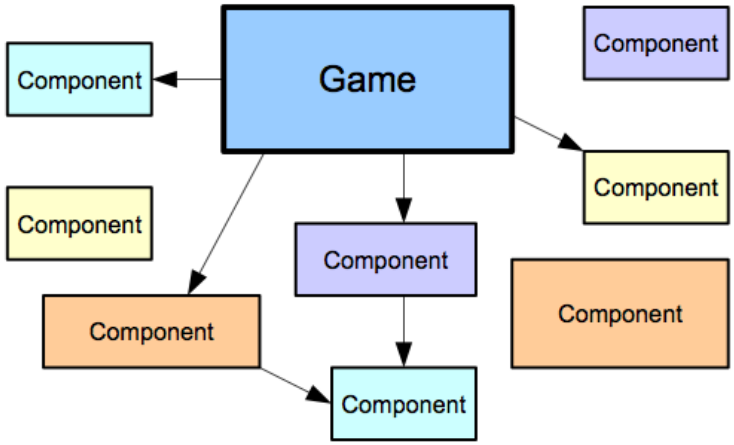


Distributed development

- Modular programming allows distributed development.
- By breaking down the problem into multiple tasks, different developers can work in parallel.
- And this will shorten the development time.

Code reusability

- A program module can be reused in programs.
- This is a convenient feature because it reduces redundant code.
- Modules can also be reused in future projects.
- It is much easier to reuse a module than recreate program logic from scratch.



Program readability

- Modular programming leads to more readable programs.
- Modules can be implemented as user-defined functions.
- A program that has plenty of functions is straightforward.
- But a program with no functions can be very long and hard to follow.

TexturedPrelitVertex.hlsl

```
    float4 pos : SV_POSITION;
    float2 texCoord : TEXCOORD0;
    float skin : TEXINDEX;
};
VertexShaderOutput main(VertexShaderInput input)
{
    VertexShaderOutput output;
    // Transform the vertex position into projected space.
    float4 worldPos = mul(float4(input.pos, 1.0f), input.mTransform);
    output.pos = mul(mul(worldPos, view), projection);
    output.texCoord = input.texCoord;
    output.skin = input.skin.x;
    return output;
}
```