Solent University

Faculty of Business, Law and Digital Technologies

**Computer Fundamentals (COM709) AE1  
Programming for Problem Solving**

Author : Jordan Da Graca, 14643472

Course Title : MSc Digital Design

Module Leader : **Prins Butt**

Module Tutor : **Andy Farnell**

Date : 12/11/2022

Table of Contents

[1 Summary of Computing Concepts 3](#_Toc124249559)

[1.1 Computing Architecture 3](#_Toc124249560)

[1.2 Storing Data 4](#_Toc124249561)

[1.3 Number Systems 4](#_Toc124249562)

[1.4 Boolean Logic 5](#_Toc124249563)

[1.5 File Structuring & Disk Storage 5](#_Toc124249564)

[1.6 Operating Systems 6](#_Toc124249565)

[2 Summary of Program Construction 7](#_Toc124249566)

[2.1 Developing Programs 7](#_Toc124249567)

[2.2 Programming Standards 8](#_Toc124249568)

[2.3 Program structure 8](#_Toc124249569)

[2.4 Version Control 11](#_Toc124249570)

[3 Designing and Developing Computer Programs 12](#_Toc124249571)

[3.1 Overall Architecture 12](#_Toc124249572)

[3.2 Modules and Functions 13](#_Toc124249573)

[4 Recommendations 14](#_Toc124249574)

[5 References 15](#_Toc124249575)

[6 Appendices 16](#_Toc124249576)

Table of Figures

[Figure 1: Data Model (Butt, P., 2021) 5](#Figure!0|sequence)

Index of Tables

## Summary of Computing Concepts

This section will detail the overarching computer concepts necessary for the completion and maintenance of a modern computer. This includes the storage, access and utilisation of data and programs.

### Computing Architecture

A computational system can be defined as “a machine that simplifies complicated tasks” (Meador, D. 2020). The general structure of a computer system consists of five key aspects, the input, output, storage, arithmetic logic, and control units. The figures below define the relationship between these units.

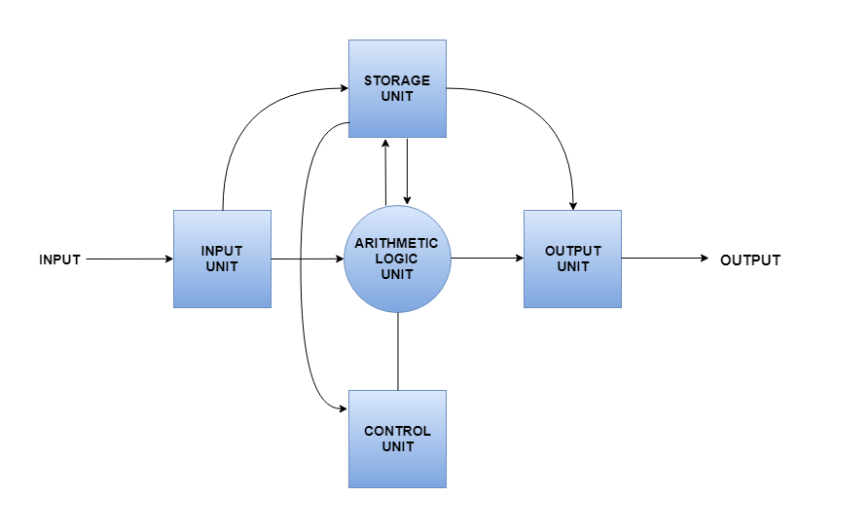


Figure 1: Basic Computer Architecture, 2020

Table 1: Core Components, 2020

|  |  |
| --- | --- |
| **Component** | **Explanation** |
| Input Unit | The input unit is responsible for the collection and conversion of data. This unit takes in data from external devices (such as keyboards or mice), converting it into usable machine language, then loads it into the system. |
| Output Unit | This provides users with the final, computed data in a readable format. Usually as audio or visual content. This can be used by monitors or headphones, for example. |
| Storage Unit | Storage units are divided into primary and secondary sources. Primary storage is directly accessed by computer components such as the CPU. Secondary units store huge amounts of data on a long-term basis. |
| Control Unit | The control unit is essentially the brain of the computer system. It converts and transfers data throughout the core computer components and dictates their behaviour. |
| Arithmetic Logic Unit (ALU) | The ALU is responsible for the completion of logical calculations with data supplied to it by the control unit. This and the control unit form the CPU. |

### Storing Data

All data stored on computers is represented as binary digits. These digits have a value of either zero or one, and strings of these can make up almost anything, including photos, documents, and videos. A byte is the most common unit of storage and is equal to 8 bits. Binary data is primarily stored in hardware storage, such as Hard Drives (HDD), Solid State Drives (SSD) or USB flash drives. Prior to these hardware options, floppy disks and tapes also stored data magnetically, however these were soon replaced, citing the facts that “Data storage density was not high, and data access times were very poor.” (McKenzie 2018)

### Number Systems

A picture containing text, white

Description automatically generatedAs mentioned previously, computers represent data as binary digits. Binary is an example of a number system, a type of representation format for numeric values digitally. Other examples of this include the decimal number system, more simply recognised as values between zero and nine. Through this, all numbers are achievable based on multiplication and squaring, however it is somewhat suboptimal compared to other methods. In addition to these two systems, the octal number system is another example of a positional value system, whereby values span between zero and eight, with higher numbers being expressed in powers of eight. The final example of a number system is the hexadecimal number system, which is an algebraic system representing 16 symbols, zero through nine, and A through F. The following table provides examples of each of these.

Table 2: Number Systems, 2022

### Boolean Logic

“Computers are made up of digital circuits. Each component in a circuit has an input voltage that can be evaluated as **True or False**, or **1 or 0**.” (ICS, 2022) Outside of this scope, a number of boolean operators are used by computers to evaluate conditional statements in search engines, algorithms, and formulas. The commonly used operators are as shown below.

|  |  |
| --- | --- |
| **Operator** | **Explanation** |
| AND | Evaluates to TRUE or 1 when both terms or conditions are evaluated. |
| OR | Evaluates to TRUE or 1 when one term or condition is evaluated. |
| NOT | Evaluates to TRUE if provided term(s) or conditions were excluded. |

Table 3: Boolean Logic, 2022

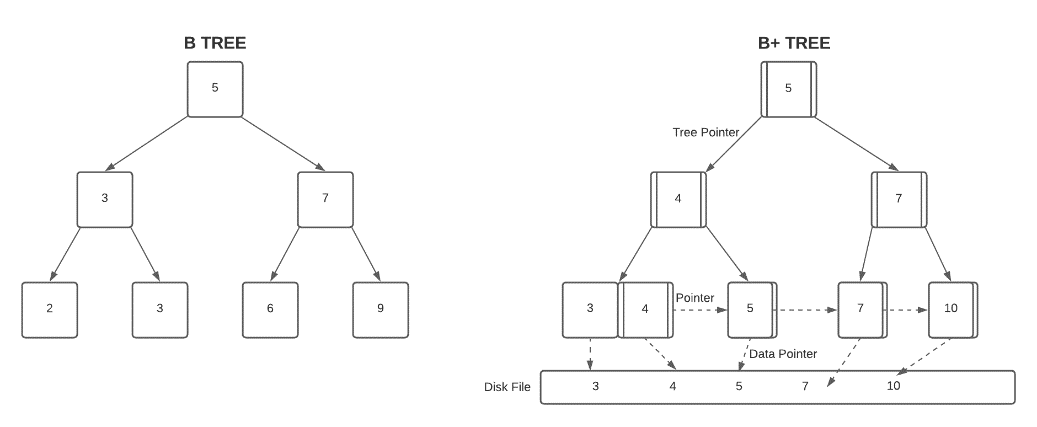
In programming, Boolean logic is often utilised in order to evaluate queries or statements, conditionally returning a value based on the status of that statement. An example of this could be a search query for a shopping application, where results can be filtered. The application of these filters would be carried out through the use of the NOT operator.

### File Structuring & Disk Storage

File structuring is a collection of operations for representing and accessing data. It enables applications such as coding IDEs (to read, write and modify data.) optimisation of file structuring is essential to ensuring applications run as efficiently as possible. As reiterated by Sari, (Sari, 2021) “A good file structure should [have] Fast access to a great capacity, reduce the number of disk accesses [and] manage growth by splitting these collections.” The following are some examples of file structures.

|  |  |
| --- | --- |
| **File Structure** | **Explanation** |
| AVL Trees | A type of binary-based search tree that incorporates dynamic self-balancing to maximise efficiency. |
| B-Trees | A b-tree is another self-balancing tree structure, responsible for maintaining sorted data. It allows for searches, access, and modifications. B-Trees are preferred for databases and file systems due to their scalability. |
| Hashing | Defined as the process of indexing and retrieving items in a data structure efficiently via the hash function. |

Table 4: File Structures, 2022



Disk Storage pertains to the utilisation of aforementioned hardware in order to store files and applications on a computer. Volatile and non-volatile storage components are used in computing. Volatile classifies anything that is only stored there until there is no longer an electrical supply (such as RAM). Non-volatile includes hard disk drives (HDD), solid state drives (SSDs) or external storage such as disks or drives.

Figure 2: B-Tree, 2022

### Operating Systems

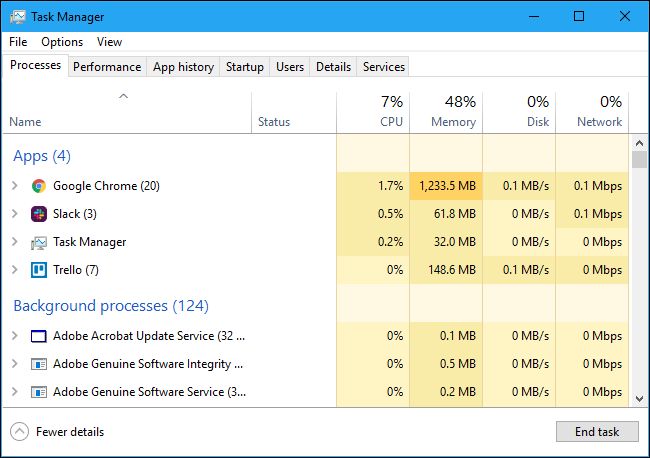
As reiterated by Cegal (Cegal, 2022) “An operating system is the most important software which runs on a computer.” It is singlehandedly responsible for controlling the computer’s memory, software, hardware and running processes. The speed of this operating system is dependant on the quality of hardware, such as the CPU, RAM and storage type (for example, a hard-drive is far slower than an M2. NVMe drive.)

Figure 3: Windows Task Manager, 2022

## Summary of Program Construction

This section summarises pre-existing and learnt behaviour regarding program construction. This includes the development lifecycle of modern computer programs, their benefits, drawbacks, and application. This section will also detail appropriate programming standards and code conventions, such as version control.

### Developing Programs

There is a plethora of computer programs that’re developed today, each with their own aim and justification for existence. An example of this is the suite of front-end web development tools, such as HTML, CSS Javascript (and its libraries). These 3 languages, whilst powerful independently, often rely on one another in order to provide developers with the necessary tools to produce captivating, modern interfaces with intuitive and programmatical interactions.

This unit provided a refresher on the popular language Python. Often seen as one of the easiest (and yet widely utilised) libraries (Coursera, 2022), Python enables developers to efficiently integrate their programs with a plethora of features, APIs and languages.

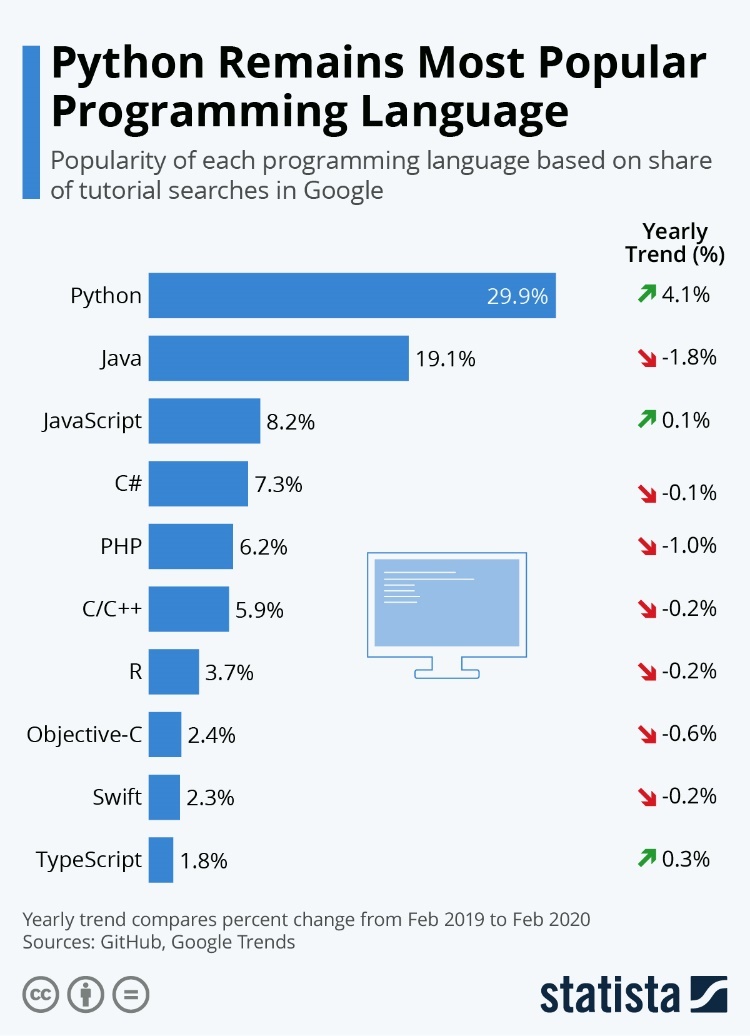


Figure 4: Popular Development Languages, 2022

### Programming Standards

There are a number of universal programming standards that ensure maximum efficiency, readability, and compliance. An example of this is a developmental aim to optimise readability for the benefit of yourself and other developers. Especially when working collaboratively, it’s important to program as efficiently as possible. This can be done through the following methods:

|  |  |
| --- | --- |
| **Convention** | **Explanation** |
| Few lines as possible | This optimises legibility, and often increases the running speed of the program. |
| Naming Conventions | Universally consistent naming conventions allows for succinct code, and reduces the room for error. |
| Segmented code | Ensuring code is spaced and indented correctly increases legibility and allows the identification of errors to become easier. |
| Work smarter not harder | Utilisation of third party packages is useful, if done in mediation. Simply copy-pasting code into your program will not benefit you in the long run. |
| DRY principle | This stands for Don’t Repeat Yourself. If something happens frequently, it is almost always better to make it a repeatable block of code, such as a function. |
| Comments | Comments allow individuals to understand the function and significance of portions of programs. |
| Version Control | Version control (such as the use of GitHub) allows developers to document the gradual evolution of their program, and revert to previous versions, should the need arise. |

Table 5: Programming Standards, 2022

### Program structure

Programs are executed linearly on the main-scope from top to bottom. Naturally this means that certain aspects must be at specific points in the program. The importation of third party packages and files (such as images) must take place immediately, therefore allowing any later code to use them unobtrusively. After this, function declarations would take place. Functions enable blocks of code to be programmatically segmented for the purpose of automation, repeatability and efficiency. The natural end of programs include the value returned to the user, or the resulting outcome of the script.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generatedText

Description automatically generated

Programming also allow for complex decision making to take place. This allows for blocks of code to be conditionally run based on IF statements. These conditions support logical conditions from mathematics, such as equals, not equals, and less / greater than. The following example conditionally returns an outcome based on whether the inputted variable is true or false.

Figure 8: Example Python Function Output, 2022

Figure 7: Example Python Function, 2022

Functions enable the automation of repeatable code. This essentially enables pre-defined workflows to be called effortlessly. Functions contain their own scope, whereby variables declared within there are local to the function. This can often create conflicts when attempting to access those variables globally. Changes made to global variables from within the scope of functions, however, are reflected globally. Below is an example of a function, whereby it is provided three parameters, and utilises those parameters to generate a string dynamically. Note that the only thing running when the script runs, is the function call.

Figure 6: Example Python Layout Output, 2022

Figure 5: Example Python Layout, 2022

Text

Description automatically generated

Text

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

Figure 11: Development Life-Cycle, 2022

Figure 10: Conditional Coding Script Outcome, 2022

Figure 9: Conditional Coding Script, 2022

### Version Control

GitHub is an industry leading version control and hosting platform industrially used to ensure the distribution of version-controlled content. This content can be pushed to various development branches of a project, allowing large teams to work on a single project simultaneously and remotely. Changes can be pushed directly from the command line and are uploaded digitally to GitHub. Below is an example of a file receiving repeated changes via GitHub version-controlled pushes.

A screen shot of a computer

Description automatically generated with low confidence

Figure 12: Version Control, 2022

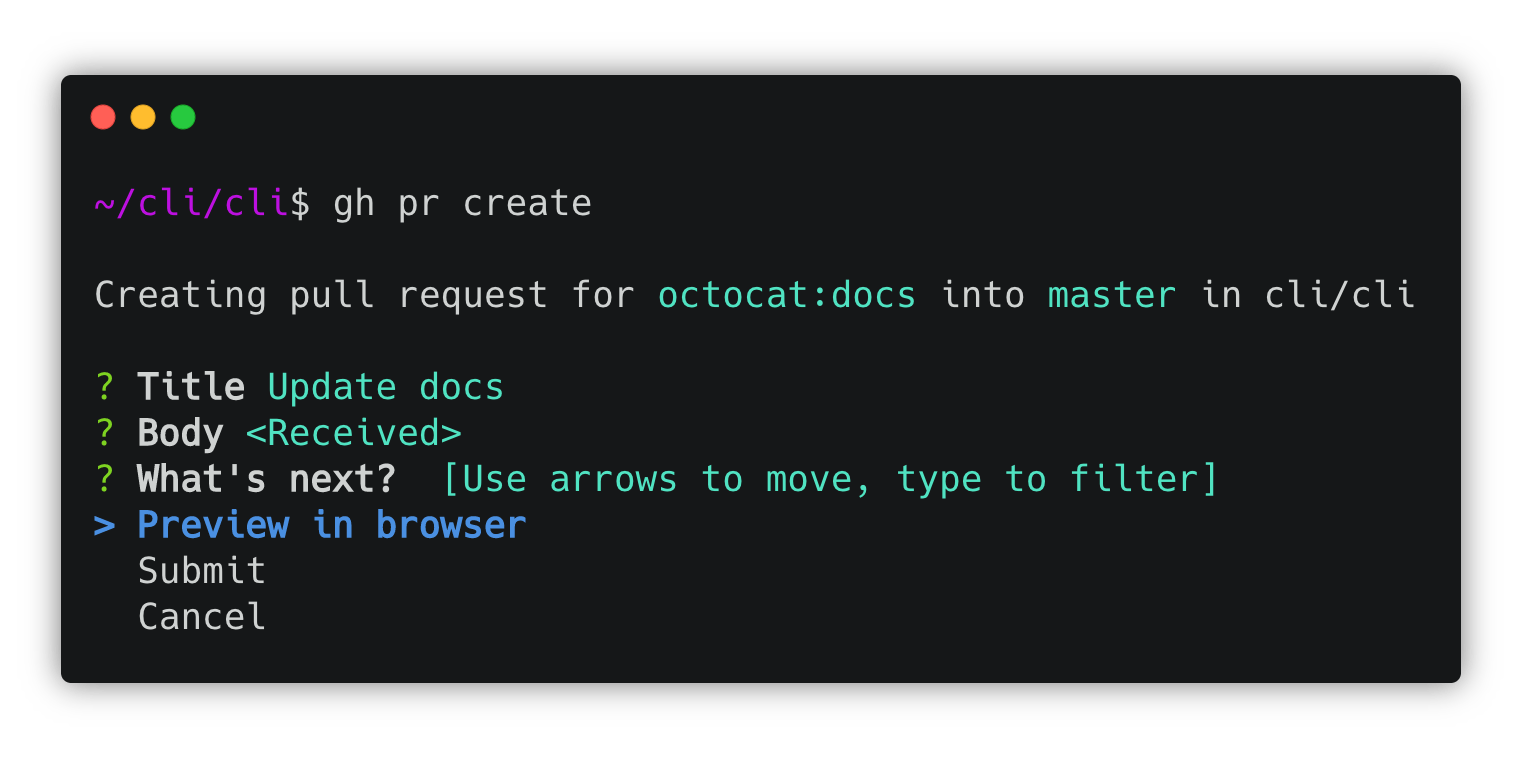
As previously mentioned, Git commands can be used remotely via the command line to ‘pull’ project files from GitHub. This allows developers to work independently and unobtrusively. As reaffirmed by Ouellette (Ouellette, 2021), “Git is an immensely popular Version Control System, which allows developers to create a history of changes that occur over time in their code.” An example of a pull-request via a command line can be seen below.

Figure 13: Pull Request, 2022

## Designing and Developing Computer Programs

A heading should not be followed directly by another heading. There should be some connecting text. Thus, you could briefly describe the structure of this section here.

### Overall Architecture

Explain, with suitable justifications, the design and implementation of your system in this section. It would be useful to include a diagram showing your system’s overall architecture e.g. a class diagram. The diagram can be included in this section as a figure or, if it is large, in the appendices and cited in this section.

You should ensure that any figures or tables are appropriately label following suitable conventions. Generally, the figure or table is centred and figure captions are placed below a figure and table captions are placed above a table.

The example below shows a figure with a caption (note, it is database model and not a class diagram).

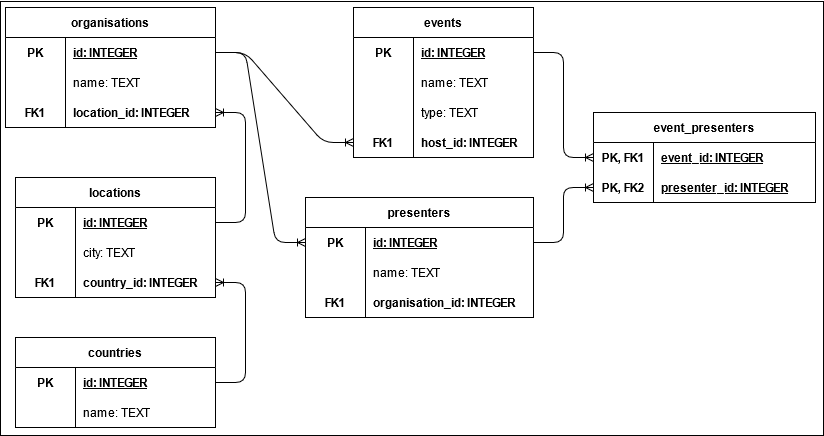


Figure 1: Data Model (Butt, P., 2021)

The example of a figure caption above also includes a reference. If an image is taken without any modifications from another source then it should include a reference as shown in the example above.

If the resource contains many pages then the page number should be included e.g.

Figure 1 – Database Model (p. 5 from Butt, P., 2021)

If the image has been adapted then this should be stated in the caption e.g.

Figure 1 – Database Model (adapted from Butt, P., 2021)

If permission has been granted to use the figure then this can also be stated in the caption e.g.

Figure 1 – Database Model (Butt, P., 2021, printed with permission)

It is important that your explanation includes the correct use of relevant terminology (e.g. parameter, function, etc.) and the application of appropriate principles (e.g. iteration). Your explanation should be clear, concise and justified. Where appropriate, you should mention alternative approaches to the design with a reasoning for your selected approach.

### Modules and Functions

It is expected that your solution will consist of multiple modules each with multiple functions. Ideally, each module should be responsible for one aspect of the system. For example, you may have a module that contains all the functions for visualisation, another with all the functions to communicate with the user, and so on.

Explain the functionality implemented in each module. Relevant guidelines mentioned in the previous section should also be followed here.

You should include suitable screenshots of the final working functionality where relevant e.g. visualisations. You can also include evidence of testing.

It may be useful to include an example or two of how different modules and functions work together to deliver implement a system function. You may, for example, present this as a diagram or a series of annotated function calls.

## Recommendations

You should briefly summarise what has been achieved and make some suitable recommendations for future work.

## References

Any references should be included here. This list should include references that have been cited in this report.

## Appendices

Any appendices should be included here.