Joint program

Algebra X Epitech

Epitech's unit

Learning Outcomes

Advanced C++ - Babel (B-CPP-500)

- Evaluate the difference between Unix and Windows operating systems when developping in C++.
- Create a working CMake file capable of building the project on both type of operating system.
- Evaluate the API of a C++ project or library to know if it fits your need.
- Design a C++ project using OOP principles.
- Produce an abstraction of sockets for Windows and UNIX.
- Write a technical documentation of the project.
- Construct a UDP protocol to transfer voice over a network.

Advanced C++ - R-Type (B-CPP-501)

- Evaluate the difference between Unix and Windows operating systems when developping in C++.
- Create a working CMake file capable of building the project on both type of operating system.
- Design a C++ project using OOP principles.
- Produce an abstraction of sockets for Windows and UNIX.
- Write a technical documentation of the project.
- Construct a correct game loop like a traditional game engine would.
- Create an Entity-Component-System to manage all entities of your game.
- Construct a network protocol to play the game over the internet.

Application Development - Dashboard (B-DEV-500)

- Consume external REST APIs from your project.
- Integrate OAuth2 authentication in your project, especially to use external APIs.
- Design widgets using data from the consumed APIs.
- Manage the update of the widgets using time.
- Explain the choices made on the user interface to offer a good User eXperience.

Application Development - Redditech (B-DEV-501)

- Consume an external REST API from your project.
- Integrate OAuth2 authentication in your project, especially to use external APIs.
- Integrate medias (picture, sound, video) correctly on your application.
- Write a technical documentation of the project.
- Identify which part of an API is valuable for the project.
- Defend the organisation used within the project group.
- Explain the choices made on the user interface to offer a good User eXperience.

Functional Programming - evalExpr (B-FUN-500)

- Apply functional programming paradigm to develop the project.
- Examine the syntax of a complex mathematical expression.
- Write a program capable of parsing primitives number.
- Implement a Parsing Expression Grammar (PEG) describing any mathematical expression.
- Construct an Abstract Syntax Tree representing a mathematical expression.

Functional Programming - HAL (B-FUN-501)

- Apply functional programming paradigm to develop the project.
- Implement a Parsing Expression Grammar (PEG) describing any mathematical expression.
- Construct an Abstract Syntax Tree representing a mathematical expression.
- Implement a read-eval-print loop (REPL) to use the program in a interactive manner.
- Demonstrate the Quality Assessment of the project.

Year-End Project (B-YEP-500)

In this unit, students can choose one of three projects.
The learning outcomes depend on the choosen project.

Zia (C++ track)

- Evaluate the difference between Unix and Windows operating systems when developping in C++.
- Create a working CMake file capable of building the project on both type of operating system.
- Design a C++ project using OOP principles.
- Produce an abstract APIs to allow extension via plugins.
- Write a technical documentation of the project.
- Implement a simple HTTP server using TCP sockets.
- Identify how to mitigate problems to keep the service running as long as possible.

AREA (Application Development track)

- Consume external REST APIs from your project.
- Integrate OAuth2 authentication in your project, especially to use external APIs.
- Write abstraction to be able to easily add Actions and Reactions.
- Write a technical documentation of the project.
- Defend the organisation used within the project group.
- Explain the choices made on the user interface to offer a good User eXperience.

KOAK (Functional Programming track)

- Apply functional programming paradigm to develop the project.
- Construct an Abstract Syntax Tree representing a mathematical expression.
- Implement a parser and lexer
- Write a program capable of inferring types

- Generate executable code using LLVM