**San Cristobal of Huamanga National University (UNSCH)**
School of Computer Science
Syllabus 2024-II

## 1. COURSE

CS112. Computer Science I (Mandatory)

## 2. GENERAL INFORMATION

| | | |
|---|---|---|
| **2.1 Course** | : | CS112. Computer Science I |
| **2.2 Semester** | : | $2^{nd}$ Semester. |
| **2.3 Credits** | : | 5 |
| **2.4 Horas** | : | 2 HT; 6 HP; |
| **2.5 Duration of the period** | : | 16 weeks |
| **2.6 Type of course** | : | Mandatory |
| **2.7 Learning modality** | : | Face to face |
| **2.8 Prerrequisites** | : | CS111. Introduction to Programming. ($1^{st}$ Sem) CS111. Introduction to Programming. ($1^{st}$ Sem) |

## 3. PROFESSORS

Meetings after coordination with the professor

## 4. INTRODUCTION TO THE COURSE

This is the second course in the sequence of introductory courses in computer science. The course will introduce students in the various topics of the area of computing such as: Algorithms, Data Structures, Software Engineering, etc.

## 5. GOALS

- Introduce the student to the foundations of the object orientation paradigm, allowing the assimilation of concepts necessary to develop information systems.

## 6. COMPETENCES

**1)** Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions. (Assessment)

**2)** Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline. (Assessment)

**5)** Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. (Familiarity)

**6)** Apply computer science theory and software development fundamentals to produce computing-based solutions. (Usage)

## 7. TOPICS

| **Unit 1: Overview of Programming Languages (1)** | |
|---|---|
| **Competences Expected: 1** | |
| **Topics** | **Learning Outcomes** |
| • Brief review of programming paradigms.<br>• Comparison between functional and imperative programming.<br>• History of programming languages (emphasis on C and C++). | • Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 2: Máquinas virtuales (2)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • The concept of a virtual machine.<br>• Tipos de virtualización (incluyendo Hardware / Software, OS, Servidor, Servicio, Red) .<br>• Intermediate languages. | • Explicar el concepto de memoria virtual y la forma cómo se realiza en hadware y software [Familiarizarse]<br>• Diferenciar emulacion y el aislamiento [Familiarizarse]<br>• Evaluar virtualización de compensaciones [Evaluar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 3: Sistemas de tipos básicos (6)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • Type systems in programming languages.<br>• Declaration models (linking, visibility, scope, and lifetime).<br>• Overview of type checking. | • Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse]<br>• Para un lenguaje con sistema de tipos estático, describir las operaciones que están prohibidas de forma estática, como pasar el tipo incorrecto de valor a una función o método [Familiarizarse]<br>• Describir ejemplos de errores de programa detectadas por un sistema de tipos [Familiarizarse]<br>• Para múltiples lenguajes de programación, identificar propiedades de un programa con verificación estática y propiedades de un programa con verificación dinámica [Usar]<br>• Dar un ejemplo de un programa que no verifique tipos en un lenguaje particular y sin embargo no tenga error cuando es ejecutado [Familiarizarse]<br>• Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar]<br>• Explicar como las reglas de tipificación definen el conjunto de operaciones que legales para un tipo [Familiarizarse]<br>• Escribir las reglas de tipo que rigen el uso de un particular tipo compuesto [Usar]<br>• Explicar por qué indecidibilidad requiere sistemas de tipo para conservadoramente aproximar el comportamiento de un programa [Familiarizarse]<br>• Definir y usar piezas de programas (tales como, funciones, clases, métodos) que usan tipos genéricos, incluyendo para colecciones [Usar]<br>• Discutir las diferencias entre, genéricos (*generics*), subtipo y sobrecarga [Familiarizarse]<br>• Explicar múltiples beneficios y limitaciones de tipificación estática en escritura, mantenimiento y depuración de un software [Familiarizarse] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 4: Conceptos Fundamentales de Programación (6)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • Diseño orientado a objetos:<br>– Descomposicion en objetos que almacenan estados y poseen comportamiento<br>– Diseño basado en jerarquia de clases para modelamiento<br>• Variables and data types.<br>• Expressions and operators.<br>• Conditional statements. | • Analiza y explica el comportamiento de programas simples que involucran estructuras fundamentales de programación variables, expresiones, asignaciones, E/S, estructuras de control, funciones, paso de parámetros, y recursividad [Evaluar]<br>• Identifica y describe el uso de tipos de datos primitivos [Familiarizarse]<br>• Escribe programas que usan tipos de datos primitivos [Usar]<br>• Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar]<br>• Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar]<br>• Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Evaluar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 5: Functions (6)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • Paso de funciones y parámetros.<br>• Parameter passing.<br>• Function overloading.<br>• Fundamentals of recursion.<br>• Function templates. | • Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar]<br>• Understand and apply the concept of parameter passing to a function, both by value and by reference. [Usar]<br>• Identify and apply the concept of function overloading. [Usar]<br>• Describe el concepto de recursividad y da ejemplos de su uso [Familiarizarse]<br>• Design, implement, and apply the concept of templates to create generic functions. [Usar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 6: Arrays, Pointers, and Memory Management (8)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • Array definition.<br>• Multidimensional arrays.<br>• Pointer fundamentals.<br>• Dynamic memory management (new/delete, stack vs. heap).<br>• Smart pointers (unique_ptr, shared_ptr, weak_ptr).<br>• Advanced pointer concepts (pointers to pointers, pointers to functions). | • Understand and implement one-dimensional arrays. [Familiarizarse]<br>• Design and apply the concept of multidimensional arrays. [Usar]<br>• Understand and apply the concept of references and pointers. [Familiarizarse]<br>• Understand, apply, and evaluate the relationship between pointers and arrays. [Evaluar]<br>• Understand and implement dynamic memory management. Differentiate between heap and stack memory regions. [Evaluar]<br>• Design, implement, and evaluate concepts like pointer-to-pointer, pointer-to-function, among other advanced pointer concepts. [Evaluar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 7: Working with Arrays and Pointers (5)** | |
|---|---|
| **Competences Expected: 1** | |
| **Topics** | **Learning Outcomes** |
| • Arrays as function arguments.<br>• Character arrays and pointers.<br>• Pointers and 2-dimensional arrays.<br>• Pointers and multidimensional arrays. | • Demonstrate the use of pointers with different types of arrays. [Usar]<br>• Demonstrate the layout of an array in memory and how pointers are manipulated within those memory spaces. [Usar]<br>• Demonstrate the use of pointer arithmetic with arrays.[Usar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 8: Pointers and Dynamic Memory (5)** | |
|---|---|
| **Competences Expected: 1** | |
| **Topics** | **Learning Outcomes** |
| • Pointers and dynamic memory - stack vs heap.<br>• Pointers as return values from a function in C/C++.<br>• Pointers to functions in C/C++.<br>• Pointers to functions and callbacks.<br>• Memory leaks in C/C++. | • Show the memory structure within a program and understand how the compiler allocates elements on the stack and heap.[Usar]<br>• Demonstrate the use of functions and operators for dynamic memory allocation and deallocation.[Usar]<br>• Understand the implications of returning pointers from functions. [Usar]<br>• Use pointers to functions as parameters. [Usar]<br>• Understand the implications of dynamic memory usage and memory leaks. [Usar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 9: Pointers and Classes (5)** | |
|---|---|
| **Competences Expected: 1** | |
| **Topics** | **Learning Outcomes** |
| • Pointers to class members - attributes.<br>• Pointers to class members - methods and calls to method pointers.<br>• Pointers to class members - static methods and calls to static method pointers.<br>• Pointers to classes - example with linked list management. | • Understand the use of pointers to different elements of a class. [Usar]<br>• Understand the use of pointers to static members of a class. [Usar]<br>• Introduce the node structure and its use in a simple data structure. [Usar]<br>• Introduce data structures, showing a simple implementation of linked lists.[Usar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 10: Programación orientada a objetos (8)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • Diseño orientado a objetos:<br>– Descomposicion en objetos que almacenan estados y poseen comportamiento<br>– Diseño basado en jerarquia de clases para modelamiento<br>• Lenguajes orientados a objetos para la encapsulación:<br>– privacidad y la visibilidad de miembros de la clase<br>– Interfaces revelan único método de firmas<br>– clases base abstractas<br>• Definición de las categorías, campos, métodos y constructores.<br>• Las subclases, herencia y método de alteración temporal.<br>• Subtipificación:<br>– Polimorfismo artículo Subtipo; upcasts implícitos en lenguajes con tipos.<br>– Noción de reemplazo de comportamiento: los subtipos de actuar como supertipos.<br>– Relación entre subtipos y la herencia.<br>• Uso de coleccion de clases, iteradores, y otros componentes de la libreria estandar.<br>• Asignación dinámica: definición de método de llamada. | • Diseñar e implementar una clase [Usar]<br>• Usar subclase para diseñar una jerarquía simple de clases que permita al código ser reusable por diferentes subclases [Usar]<br>• Razonar correctamente sobre el flujo de control en un programa mediante el envío dinámico [Usar]<br>• Comparar y contrastar (1) el enfoque procedurar/funcional- definiendo una función por cada operación con el cuerdo de la función proporcionando un caso por cada variación de dato - y (2) el enfoque orientado a objetos - definiendo una clase por cada variación de dato con la definición de la clase proporcionando un método por cada operación. Entender ambos enfoques como una definición de variaciones y operaciones de una matriz [Evaluar]<br>• Explicar la relación entre la herencia orientada a objetos (codigo compartido y *overriding*) y subtipificación (la idea de un subtipo es ser utilizable en un contexto en el que espera al supertipo) [Familiarizarse]<br>• Usar mecanismos de encapsulación orientada a objetos, tal como interfaces y miembros privados [Usar]<br>• Definir y usar iteradores y otras operaciones sobre agregaciones, incluyendo operaciones que tienen funciones como argumentos, en múltiples lenguajes de programación, selecionar la forma mas natural por cada lenguaje [Usar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 11: Templates and STL (6)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • Class templates.<br>• Basic concepts of the Standard Template Library (STL) including: vector, list, stack, queue. | • Understand the concepts of class templates. [Familiarizarse]<br>• Implement and create new generic data types. [Usar]<br>• Understand the basic structures of the STL. [Familiarizarse]<br>• Use basic data structures like stack, queue, list, and vector from the STL. [Usar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 12: Operator Overloading (4)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • Operator overloading definition. | • Understand the concepts of operator overloading. [Familiarizarse]<br>• Implement the overloading of allowed operators in the programming language. [Usar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

| **Unit 13: File Handling (4)** | |
|---|---|
| **Competences Expected: 1,6** | |
| **Topics** | **Learning Outcomes** |
| • File input and output (I/O). | • Understand the concepts of file manipulation. [Familiarizarse]<br>• Create programs to read and write files. [Usar] |
| **Readings :** [Str13], [Jos19], [Dei17] | |

## 8. WORKPLAN

### 8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

### 8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

### 8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

## 9. EVALUATION SYSTEM

********* EVALUATION MISSING ********

## 10. BASIC BIBLIOGRAPHY


[Str13] Bjarne Stroustrup. *The C++ Programming Language*. 4th. Addison-Wesley, 2013.

[Dei17] Deitel & Deitel. *C++17 - The Complete Guide*. 10th. Pearson, 2017.

[Jos19] Nicolai M. Josuttis. *C++17 - The Complete Guide*. 1st. 2019.