LAB II	Profesor: Ruben Calabuig
TP Integrador Nº 02	OOP: Videojuego.

FICHA TÉCNICA

EDAD DE LOS ALUMNOS: Alumnos del 2º cuatrimestre de TSP.

COMPETENCIAS

Para desarrollar el TP, deberás tener conocimientos previos de los siguientes temas:

- Programación básica en lenguaje C++.
- Programación modular y estructurada.
- Programación orientada al objeto (OOP).

Al finalizar la tarea deberás ser capaz de:

- Comprender los fundamentos de la OOP.
- Diseñar y administrar clases y objetos.
- Realizar un un juego utilizando el paradigma OOP de mediana complejidad.

TAREA

Durante el desarrollo de la tarea deberás diseñar y desarrollar un videojuego utilizando como paradigma OOP.

NOTA:
Se recomienda que se realice un videojuego simple de acción, azar, lógica, etc.

CONSIGNAS

- **CONSIGNAS PARA EL DESARROLLO**
  - Lee atentamente todas las consignas antes de iniciar la tarea.
  - El alumno debe entregar una propuesta de video juego basado en las consignas antes de comenzar el desarrollo del video juego.
  - Para llevar a cabo la tarea, te sugiero que dividas el proceso en etapas:
    - El programa debe ser desarrollado en C++ utilizando como entorno de desarrollo el IDE Code::Blocks, con el compilador GNU HYPERLINK "http://es.wikipedia.org/wiki/GNU_Compiler_Collection" gcc.
    - El videojuego debe ser desarrollado en su totalidad en modo gráfico. Para tal fin, la cátedra suministrará un sistema de clases basadas en las librerías gráficas SDL (Simple DirectMedia Layer) de Sam Lantinga, que se distribuyen bajo licencia GPL (General Public License).

- El paradigma de programación a utilizarse para el desarrollo del videojuego es la **Programación Orientada a Objetos (OOP)**.

NOTA:
Todo el desarrollo debe estar basado en clases, las que se definirán en archivos independientes tipo “.h” y “.cpp”, los que serán incluidos en el archivo que contiene a la función main().

- La interfase del usuario debe presentar las siguientes características:
  - El videojuego debe comenzar con una pantalla de presentación y terminar con pantalla de despedida.
  - El control del programa debe realizarse a través de un sistema de menú.
  - Se debe tener la posibilidad de guardar y recuperar un juego iniciado o debe generar, mantener y mostrar un archivo de los 10 mejores puntajes.
  - Debe existir la posibilidad de salir del programa en cualquier momento de su ejecución.
    - Según el momento de ocurrida la petición de abandono, pueden presentarse diferentes situaciones que se detallan a continuación:
      - Si se presionó la opción “salir” del Menú Principal, debe terminarse el programa.
      - Si se presionó la opción “salir”, la tecla “ESC” o se realizó alguna acción equivalente, desde algún submenú, debe retornarse al Menú Principal.

        Si se presionó la tecla “ESC” o se realizó alguna acción equivalente durante la ejecución del videojuego debe pedirse confirmación, ofrecerse la posibilidad de guardar el juego iniciado y retornar al Menú Principal.

NOTA:
Los alumnos podrán realizar propuestas de mejoras sobre estos tópicos.

• CONSIGNAS PARA LOS ARCHIVOS FUENTES

- El proyecto debe ser entregado con todos sus archivos fuentes (programas y librerías).
- Los archivos fuentes deben presentar el encabezado de autor, y cada una de las funciones debe, también, presentar un encabezado explicativo.

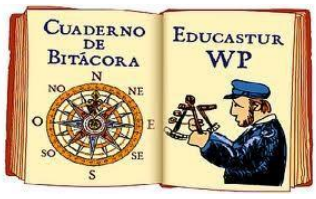
PROCESO

Etapa 1: Organización

¡Organízate! Sin orden no podrás alcanzar los objetivos.

Recuerda que tienes pocas clases para entregar tu producción.

- Lleva un cuaderno de bitácora donde registres todos los datos que te resulten relevantes para el desarrollo de la tarea (ideas que se te vayan ocurriendo, etc.).



- Crea una carpeta en tu disco, donde puedas guardar todo el material que consideres necesario para tu sistema.

Etapa 2: Diseño de la interfase

No te apresures, piensa.

- Dedica tiempo a ordenar tus ideas.
- Lee bien esta guía, y asegúrate de haber entendido bien todo lo que se te pide.

- Anota en la bitácora todo lo que se te vaya ocurriendo.
- Asígnale niveles jerárquicos a lo que vayas anotando.
- Repasa lo escrito.
- En papel haz un esquema de los niveles de menú que necesitas, y de las opciones que figuran en cada uno de ellos.

• Etapa 3: Prototipo del sistema de menús

- Prepara en Code:Blocks un prototipo del sistema de menús.
- Pruébalo y corrígelo hasta que veas que funciona aceptablemente.

- NO TE OLVIDES DE:
  - Validar todos los ingresos.
  - Trabajar con módulos pequeños.
  - Poner cada clase en una librería independiente.
  - Documentar todos los archivos, clases y métodos.

• Etapa 4: Diseño de las clases

- Revisa nuevamente los requisitos datos mínimos y relaciónalos con los objetos que formarán parte de tu videojuego.
- Diseña las clases que creas necesarias.

- RECUERDA QUE
  - Las características y compartimientos que tiene una persona, no son las mismas que las que tienen una piedra, un árbol, un perro, un avión, un contador de vueltas o un medidor de tiempo (***Piensa en objetos no en funciones***).

• Etapa 5: Diseño de las entradas y salidas

- Prepara en Code:Blocks un prototipo de las pantallas de ingresos de datos y salida de información (*listados, ayudas, ingresos de datos, mensajes de error, indicadores,* etc.)
- Pruébalo y corrígelo hasta que veas que funciona aceptablemente.

- NO TE OLVIDES DE:
  - Validar todos los ingresos.
  - Trabajar con módulos pequeños.
  - Poner cada clase en una librería independiente.
  - Documentar todos los archivos, clases y métodos.

- **Etapa 6: Diseño de los archivos**
  - Revisa nuevamente los requisitos datos mínimos.
  - Diseña los archivos que creas necesarios.

- RECUERDA QUE
  - No es lo mismo guardar un listado de nombres de personas con sus mejores puntajes, que guardar todo los datos que necesitas para reanudar un juego suspendido.

- **Etapa 7: Diseño de las ayudas**
  - Prepara en Code:Blocks un prototipo de los objetos que te permitan obtener ayuda en todo momento.
  - Pruébalo y corrígelo hasta que veas que funciona aceptablemente.

- NO TE OLVIDES DE:
  - Validar todos los ingresos.
  - Trabajar con módulos pequeños.
  - Poner cada clase en una librería independiente.
  - Documentar todos los archivos, clases y métodos.

- **Etapa 8: Integración**
  - Integra en Code:Blocks todos los prototipos desarrollados en las etapas anteriores.
  - Prueba y ajusta la integración.

- NO TE OLVIDES DE:
  - Validar todos los ingresos.
  - Trabajar con módulos pequeños.
  - Poner cada clase en una librería independiente.
  - Documentar todos los archivos, clases y métodos.

RECURSOS

 Esta Guía: "Lab2 – Trabajo Práctico Integrador 1 – Pautas – 2011.odt”

 Curso de Cpp - Pozo Coronado.pdf

 Introducción a la OOP - Grupo EIDOS.pdf

LAB II	Profesor: Rubén Calabuig
TP Integrador Nº 02	OOP: Videojuego.

Sitio de descarga gratuita de LibreOffice Org en español .

C++ con clase → Excelente sitio de programación en C y C++ de Salvador Pozo Coronado.

Curso de C++ → Libro de C++ para descargar --> Autor: Salvador Pozo Coronado.

Code:Blocks →IDE multiplataforma y multilenguaje Open Source. Sitio Oficial.

Librerías SDL → Conjunto de librerías gráficas diseñadas por Sam Lantinga.

Adn++ → Sistema de clases diseñadas a partir de las librerías SDL por Rubén Calabuig.

Foxti Reader: Lector de archivos PDF (*Portable Document Format*), es una gran alternativa, aunque su licencia no es libre, sino freeware.

7Zip → Programa libre para la compresión de archivos tipo ZIP

Computadora

Acceso a Internet

PACIENCIA

Y

SOBRE TODO