

Universidad Tecnológica Centroamericana  
UNITEC

9 de agosto del 2019

**Sesión #4**

Laboratorio de Programación III

Javier Cano

# Desarrollo de la práctica

Para el desarrollo satisfactorio del laboratorio, siga al pie de la letra cada instrucción que a continuación se le presenta.

# Objetivos del laboratorio

* Modelar soluciones a problemas mediante uso de programación orientada a objetos en C++.
* Hacer uso de herencia y polimorfismo.

# Ejercicio practico

El año es 2012, y usted decide crear un juego incremental que corre en líneas de comando, llamado **Cookie Clicker**.

El objetivo del juego es simple, consiste en acumular el mayor número de galletas posibles, cuando las únicas acciones que puede realizar el jugador son **comprar edificios o mejoras (con galletas)** y hacer **clic para producir galletas**.

El jugador posee una **repostería** con toda su información básica (nombre del jugador y numero de galletas que tiene, empieza con 0). Adentro de la **repostería**, se lleva control de todos los edificios que posee el jugador. Existen tres tipos diferentes de edificios:

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Precio Base** | **Producción Base** | **Aumento (de precio)** |
| Granja | 50 | 5 | 15 |
| Banco | 100 | 15 | 40 |
| Templo | 200 | 30 | 100 |

De igual manera que los edificios, las **reposterías** pueden tener **mejoras**. Existen dos mejoras:

|  |  |
| --- | --- |
| **Nombre** | **Precio** |
| Oreo | 100 |
| Migajas | 1000 |

El jugador puede comprar cuantos edificios quiera (siempre y cuando tenga el número de galletas suficiente para comprarlo), y cada vez que el jugador compra un edificio su precio aumenta dependiendo del valor de aumento del edificio, por ejemplo:

|  |  |
| --- | --- |
| **Templos Comprados** | **Precio del siguiente templo** |
| 0 | 200 |
| 1 | 300 |
| 2 | 400 |

El jugador también puede comprar mejoras, pero las mejoras **solo se pueden comprar una sola vez.**

Además de comprar edificios o mejoras, el jugador también puede “hacer clic” para producir galletas. Inicialmente (ya que el jugador empieza sin mejoras ni edificios), cada vez que el jugador hace clic, solo gana una galleta. Cuando el jugador empieza a comprar edificios, cada edificio contribuye en producción de galletas cuando el jugador hace clic, por ejemplo:

|  |  |  |  |
| --- | --- | --- | --- |
| **Numero de clic** | **Templos del jugador** | **Granjas del jugador** | **Galletas del jugador** |
| 0 | 1 | 2 | 00 |
| 1 | 1 | 2 | 00 + 30 + 5 + 5 + 1 |
| 2 | 1 | 2 | 41 + 30 + 5 + 5 + 1 |

Además de lo mencionado anteriormente, las mejoras y edificios son **polimórficos** y tienen características especiales que los hacen únicos:

|  |  |
| --- | --- |
| **Edificio** | **Característica** |
| Granja | Cada 10 veces que produce, otorga 20 galletas extra |
| Banco | Cada vez que produce, gana +1 de producción base para el banco |
| Templo | Cada vez que produce, gana +3 de producción base para el templo |

Cabe destacar que estas características aplican para cada uno de los edificios que tiene el jugador.

Lo mismo aplica para las mejoras:

|  |  |
| --- | --- |
| **Mejora** | **Característica** |
| Oreo | Aumenta la producción en el doble |
| Migajas | Aumenta la producción en el triple |

Su tarea es implementar el juego según estas especificaciones, donde al principio del juego se le pide el nombre del jugador, y una vez se entra al menú principal, que debe cumplir con los siguientes requisitos:

Repostería de Marcos, tiene 5493 galletas:

1. Hacer clic
2. Comprar Granja
3. Comprar Banco
4. Comprar Templo
5. Comprar Oreo (si no la ha comprado)
6. Comprar Migajas (si no la ha comprado)
7. Mostrar edificios y mejoras compradas

# Ponderación

|  |  |
| --- | --- |
| Elemento | Puntaje |
| Resolución del problema | 4 |
| Estructura de clases correcta | 3 |
| Uso correcto de polimorfismo | 2 |
| Uso correcto de herencia | 1 |

# Comentarios adicionales

Subir el documento como un archivo de **Word (.docx),** incluyendo su nombre completo, número de cuenta y repositorio de GitHub.

**Asegúrense de probar su programa y de que el código sea legible.**

Se aplicarán penalizaciones por quiebres en el código, interrupciones o errores inesperados en el programa y por mala indentación del código. Por cada una de las penalizaciones encontradas se les restara un punto de la nota final de su laboratorio.

Si el código no es legible para otras personas o está mal indentado, entonces los instructores no nos molestaremos por tratar de entender que hace su programa en caso de que se presente un error en el mismo.