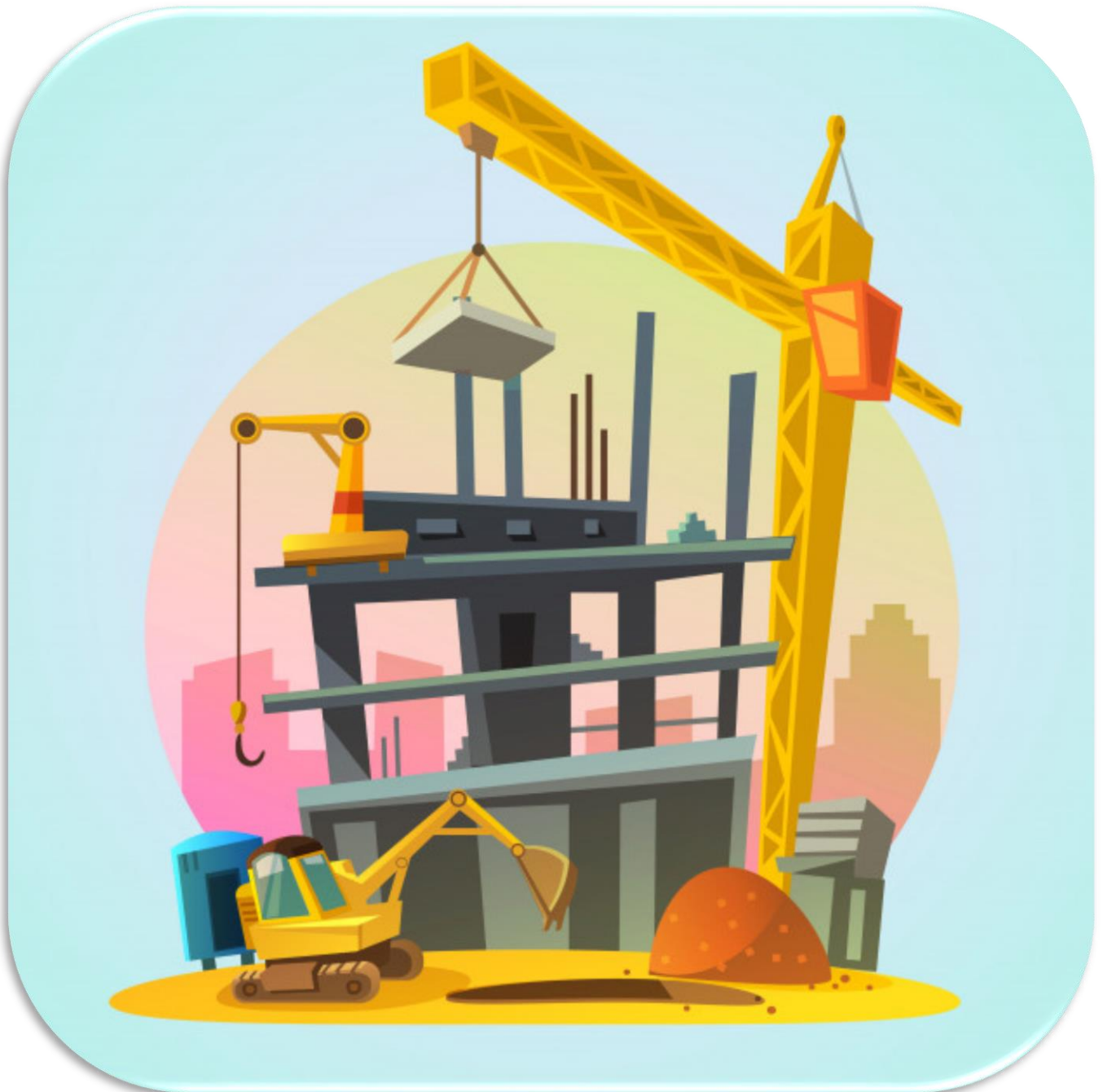


# Andypolis: la construcción

Algoritmos y programación II - 2C2021

Trabajo práctico Nº2

Grupal



## Introducción

---

Luego de muchos años de espera Andy finalmente encontró el lugar perfecto para asentarse y comenzar a armar su pequeño paraíso. Como primera medida decidió nombrar su nuevo hogar como Andypolis la maravillosa ciudad de los programadores.

Poco a poco con el pasar de los días el rumor del surgimiento de una ciudad exclusiva para programadores se fue difundiendo y la gente comenzó a llegar a ella y se armó un programa que ayude a contabilizar los materiales y edificios de la ciudad.

Fue tanta la popularidad de este programa que llegaron aun mas programadores a ayudar y pasar el diseño de Andypolis al programa, pero algunos de ellos, fanáticos de los videojuegos, decidieron darle una pequeña vuelta de tuerca...

Pasaron unas semanas y los habitantes de Andypolis nos pidieron ayuda para generar una nueva versión de nuestro programa. Como la dificultad del programa aumento nos pidieron que dos programadores se unan y desarrollen su programa en conjunto.

## Enunciado

---

Para esta nueva parte del desarrollo se continuarán usando los 2 archivos de mencionados en la parte uno del trabajo practico. Se deberá usar memoria dinámica para cargar los datos de los archivos.

- **materiales.txt**: mantendrá el formato del trabajo practico anterior.
- **edificios.txt**: ya **no tendrá la cantidad de edificios construidos**. Su nuevo formato será el siguiente:

**nombre\_edificio piedra madera metal máxima\_cantidad\_permitidos**

Por ejemplo:

```
aserradero 30 100 50 2
fabrica 100 100 250 4
escuela 250 200 45 1
yacimiento 125 140 0 2
...
```

**NOTA:** no hay límite para los edificios<sup>1</sup> ni materiales<sup>2</sup> se deberán leer ambos archivos hasta el final y eso determinara cuantos materiales y edificios existen.

Para esta nueva parte contaremos con dos archivos nuevos. En primer lugar, un archivo que nos indicará donde están ubicados los edificios construidos hasta el momento, **ubicaciones.txt**<sup>3</sup>. Este archivo tendrá el siguiente formato:

**nombre\_edificio (fila, columna)**

Por ejemplo:

```
oblisco (2, 3)
escuela (1, 6)
escuela (4, 2)
yacimiento (4, 8)
...
```

Por otro lado, tendremos un archivo **mapa.txt** que nos indicara los tipos de terrenos<sup>4</sup> que tenemos en Andypolis. Este archivo seguirá el formato:

```
cantidad_filas cantidad_columnas
tipo_terreno tipo_terreno ... tipo_terreno
.....
tipo_terreno tipo_terreno ... tipo_terreno
```

Por ejemplo:

---

<sup>1</sup> Como mínimo hay un edificio en el archivo. Se les asegura que el archivo tiene por lo menos un edificio no se requiere comprobación.

<sup>2</sup> Como mínimo existen la piedra, el metal y la madera. No se requiere comprobar que estén en el archivo, se les asegura que siempre estarán y tendrán como mínimo 0 unidades cuando no se tiene ese material (ver ejemplo de materiales.txt con madera).

<sup>3</sup> Se asegura que cualquier edificio que se encuentre en este archivo existe en el archivo edificios.txt.

<sup>4</sup> Ver anexo mapa.

8 10

LLTTTCCTTT

LTTTTCCTT

LLCCTCCTTT

CLTTTCCTTL

TLTTTLCTTC

TTTTTCCTTT

CCTCTCCTTT

TTTTTCCTTC

Una vez obtenidos los datos de los archivos armar un programa que cuente con el siguiente menú:

1. Construir edificio por nombre.
2. Listar los edificios construidos.
3. Listar todos los edificios.
4. Demoler un edificio por coordenada.
5. Mostrar mapa.
6. Consultar coordenada.
7. Mostrar inventario.
8. Recolectar recursos producidos.
9. Lluvia de recursos.
10. Guardar y salir.

### *Construir edificios por nombre*

Se deberá verificar que exista el edificio, se cuente con la cantidad de materiales necesaria para poder construir y que no se haya superado el máximo de construcciones permitidas del mismo. Si no cumple dichas condiciones se le avisara porque no es posible construir el edificio pedido, en caso contrario, se le deberá consultar al usuario si desea o no construir el edificio.

Si el usuario desea construir el edificio deberá indicar las coordenadas donde desea hacerlo. Se le deberá indicar al usuario si se construyó el edificio o si no fue posible hacerlo ya sea porque puso coordenadas fuera del mapa o indicó una posición en la que no se puede construir.

### *Listar los edificios construidos*

Se deberán listar todos los edificios **construidos**, es decir, que haya por lo menos un edificio de este tipo indicando cuantos hay construidos de cada uno y las coordenadas en las que se encuentran.

### Listar todos los edificios

Se deberán listar todos los edificios indicando para cada uno de ellos: cuantas unidades de cada material se requieren para construir uno, cuantos fueron construidos hasta el momento, cuantos más puedo construir<sup>5</sup> sin superar el máximo permitido y si me brinda algún tipo de material<sup>6</sup>.

### Demoler un edificio por coordenada

Se le pedirá al usuario que ingrese las coordenadas del edificio que desea demoler, en caso de que haya un edificio en esa posición se demolerá y se devolverán la mitad de los materiales utilizados para su construcción.

### Mostrar mapa

Mostrara el mapa indicando los edificios y materiales que se encuentren en el mismo. También se deberá mostrar ya sea con el código de colores sugerido en la sección mapa o imprimiendo con las letras indicadas los tipos de terrenos presentes en el mapa.

### Consultar coordenada

Se le pedirá al usuario que ingrese una coordenada y si la misma es válida obtendrá un mensaje con información sobre la coordenada seleccionada.

Si no hay nada en ese casillero se deberá decir que tipo de casillero es y que está vacío. Por ejemplo:

*“Soy un casillero construible y me encuentro vacío.”*

Si hay algo el casillero deberá decir que tipo de casillero es y además se le deberá pedir al objeto que de información de si mismo. Por ejemplo, si hay una piedra en un camino se podría mostrar un mensaje como el siguiente:

*“Soy un casillero transitable y no me encuentro vacío.”*

*Soy una piedra y me encuentro en el casillero consultado”*

**NOTA:** El casillero deberá mostrar su información y pedirle al objeto que contiene que muestre la suya.

### Mostrar inventario

Se deberá mostrar el inventario del jugador, es decir, los materiales que tiene el mismo. Es el equivalente a la opción listar materiales del trabajo práctico 1.

### Recolectar recursos producidos

Se recolectarán todos los materiales que produjeron los edificios que se encuentran construidos.

### Lluvia de recursos

Se generarán alrededor del mapa, en los casilleros permitidos, madera, piedra y metal. Se podrá tener como mucho 1 unidad de cualquier material por casillero, **sin permitir superposición** y siguiendo la siguiente lógica:

- Se generarán entre 1 y 2 piedras de forma aleatoria.

---

<sup>5</sup> Entiéndase por poder construir no superar el máximo de permitidos en esta parte no es necesario verificar los materiales.

<sup>6</sup> Ver anexo edificios

- Se generarán entre 0 y 1 maderas de forma aleatoria.
- Se generarán entre 2 y 4 metales de forma aleatoria.

**NOTA:** La cantidad a generar de cada material se establece de forma aleatoria, respetando los límites establecidos. Es decir, no puedo generar menos de 1 piedra ni más de 2. La ubicación en la que se genere el material también es aleatoria y deberá respetar el tamaño del mapa.







**NOTA2:** No es obligatorio guardar los materiales que lluevan, si deciden hacerlo obtendrán un punto extra y deberán agregarlos en ubicaciones.txt respetando el formato de dicho archivo. De no desear hacerlo todos los materiales que se encuentren en el mapa se perderán al cerrar el juego.

### Guardar y salir

Al terminar de usar el programa se deberán actualizar todos los archivos con los valores correspondientes manteniendo el formato de cada uno de ellos.

## Edificios

Los edificios brindaran los siguientes materiales cada vez que se pida recolectar materiales:

-  Mina: brinda 15 piedras.
-  Aserradero: brinda 25 maderas.
-  Fabrica: brinda 40 metales.
-  Escuela: no brinda materiales.
-  Obelisco: no brinda materiales.
-  Planta eléctrica: no brinda materiales.

## Mapa

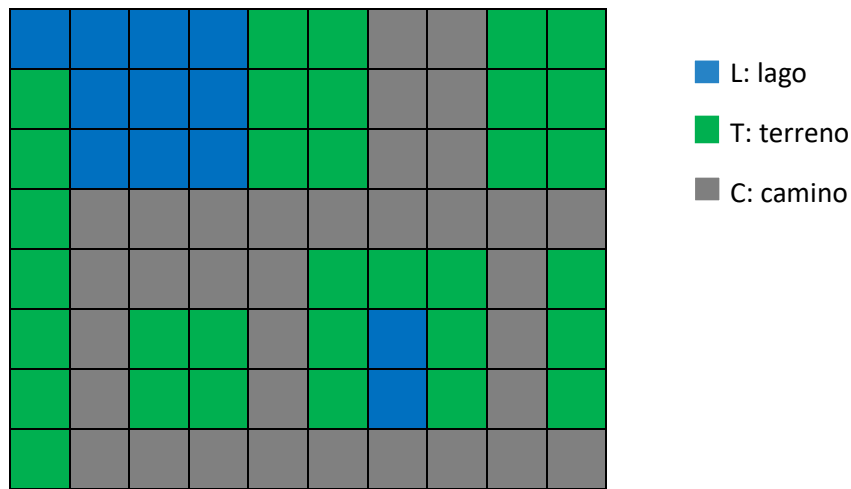
El mapa deberá implementarse con una matriz dinámica y polimórfica.

Para identificar los materiales y edificios en el mapa se usarán los siguientes caracteres:

- Mina: M
- Aserradero: A
- Fabrica: F
- Escuela: E
- Obelisco: O
- Planta eléctrica: P
- Madera: W
- Piedra: S
- Metal: I

El mapa no podrá tener nunca dos materiales o edificios en la misma posición.

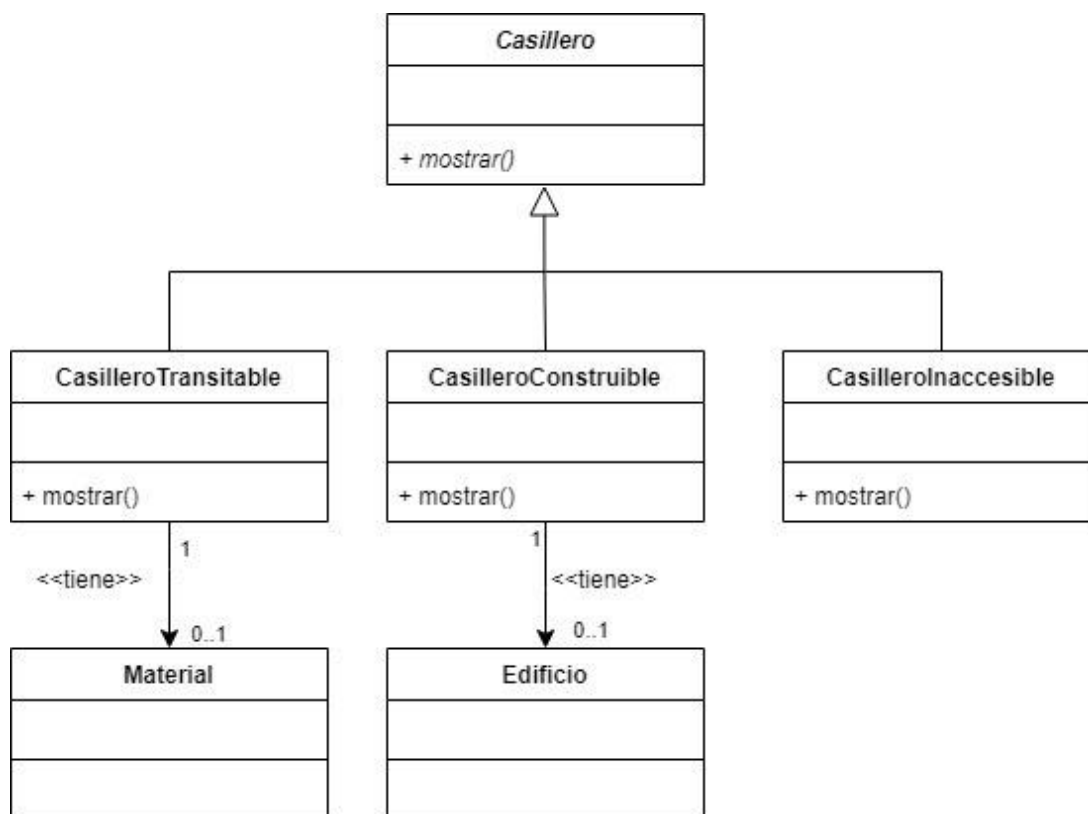
Los terrenos se podrán identificar mediante colores o caracteres de la siguiente manera:



Dependiendo el terreno podremos o no ubicar algo en esa posición de forma tal que

- Los **lagos** **no** serán **accesibles**. Pertenecerán a los casilleros inaccesibles.
- En los **terrenos** solo podrán ubicarse **edificios**. Pertenecerán a los casilleros construibles.
- En los **caminos** solo podrán ubicarse **materiales**. Pertenecerán a los casilleros transitables.

Esta implementación deberá ser hecha respetando el siguiente diagrama:



## Aclaraciones

---

- ✚ El trabajo es de carácter grupal, los grupos estarán compuestos por dos integrantes.
- ✚ Cada grupo deberá decidir cual los trabajos prácticos de los integrantes usar como base o si fusionar los mismos. Todos los errores marcados en la corrección del trabajo practico 1 deberán ser corregidos para esta segunda parte.
- ✚ El trabajo no cuenta con reentrega.
- ✚ Los archivos están bien formados.
- ✚ No recorrer múltiples veces los archivos innecesariamente.
- ✚ Se deben validar los datos ingresados por el usuario.
- ✚ No se deben subir archivos de configuración de los IDEs (**solo subir .cpp y .h**).
- ✚ El trabajo debe compilar con los flags -Wall -Werror -Wconversion.
- ✚ No se permiten usar bibliotecas de templates como por ejemplo STL.
- ✚ Se deberá usar Github, Gitlab o alguna otra plataforma en la que puedan generar un repositorio al cual subir su código y permita ver las estadísticas de dicho repositorio. Se tendrá en cuenta la participación de cada individuo en el repositorio. Además, el repositorio deberá ser privado.

## ¿Qué se evaluará?

---

- ✚ Compilación.
- ✚ Funcionalidad.
- ✚ Eficiencia espacial.
- ✚ Eficiencia temporal.
- ✚ Buenas prácticas de programación (nombres descriptivos, indentación, etc.)
- ✚ Modularización.
- ✚ Precondiciones y postcondiciones.
- ✚ Participación individual.
- ✚ Utilización de memoria dinámica.
- ✚ Utilización de polimorfismo, en especial para el mapa y el guardado de los edificios.
- ✚ Utilización de herencia, en especial para los edificios.
- ✚ Correcta utilización de objetos.



## *Normas de entrega*

---

Se deberá subir al campus un único archivo comprimido (.zip o .rar) en la sección TPs. Este archivo deberá tener un nombre formado de la siguiente manera:

**NombreGrupo\_TP2**

Por ejemplo:

duo\_dinamico\_TP2.zip

Deberá contener solo los archivos fuente. Es decir, solo .cpp y .h. NO subir los archivos de configuración de sus IDEs. (por ejemplo: CMakeList y cmake-build para Clion, .vscode para VisualStudioCode).

La fecha de **entrega** vence el **lunes 8/11/2021** a las 23.55hs.

**Puntaje:** 35 puntos.