

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 aún más 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.

Fue tal la popularidad del programa que se decidió armar un juego de 2 jugadores basado en esta gran ciudad. Como la dificultad del programa aumentó nos pidieron que cuatro programadores se unan y desarrollen el juego en conjunto.

Enunciado

Para este trabajo práctico se continuará con el desarrollo de un juego sobre construcción para dos jugadores. Los edificios se encontrarán en un diccionario¹. El diccionario se irá construyendo a medida que se lee un archivo .txt con el nombre **edificios.txt** tal y como se hacía en la parte 2 del trabajo práctico.

Para esta nueva parte del desarrollo se continuarán usando los 4 archivos de mencionados en la parte dos del trabajo practico.

- **materiales.txt**: presenta modificaciones.
- **edificios.txt**: mantendrá el formato del trabajo practico anterior.
- **ubicaciones.txt**: presenta modificaciones.
- **mapa.txt**: mantendrá el formato del trabajo practico anterior.

Cambios en materiales.txt

Al tener dos jugadores debemos indicar cuantos materiales tiene cada uno por lo que el archivo de materiales pasará a tener el formato:

nombre_material cantidad1 cantidad2

¹ Ver sección Diccionario

En el que cantidad1 es la cantidad del material que posee en su inventario el jugador 1 y cantidad2 es la cantidad de ese material que posee en el inventario el jugador 2.

Ejemplo:

```
madera 150 0
piedra 0 0
metal 200 814
andycoins 15 1000
bombas 1 0
```

Notaran que contamos con dos nuevos materiales andycoins y bombas, los cuales se explicaran en detalle más adelante. Estos 5 materiales mostrados en el ejemplo estarán siempre presentes en el archivo, aunque su valor sea 0 (ejemplo la piedra), es importante aclarar **que no aseguramos el orden de aparición de estos.**

NOTA: no deberán cambiar la lógica de lectura, se debe seguir haciendo con memoria dinámica y asumiendo que no saben la cantidad de líneas del archivo.

Cambios en ubicaciones.txt

Como mencionamos anteriormente ahora contamos con 2 jugadores por lo que usaremos el 1 y 2 como delimitadores. Además, es obligatorio guardar los materiales que estén en el mapa en este archivo. Por lo tanto, el nuevo formato del archivo será:

```
nombre_material (fila, columna)
nombre_material (fila, columna)
....
1 (fila, columna)
nombre_edificio (fila, columna)
nombre_edificio (fila, columna)
.....
2 (fila, columna)
nombre_edificio (fila, columna)
nombre_edificio (fila, columna)
...
```

Las primeras líneas serán para los materiales, luego se encontrará un 1 con la fila y columna en la que se encuentra el jugador 1, las líneas siguientes indicarán los edificios del jugador 1. Por último, se encontrará un 2 con la fila y columna en la que se encuentra y jugador 2 y en las líneas siguientes encontraremos lo edificios del jugador 2.

Por ejemplo:

```
madera (1, 0)
1 (7, 6)
escuela (1, 6)
escuela (4, 2)
mina (4, 8)
2 (2, 7)
mina oro (4, 6)
fabrica (8, 0)
```

Partida nueva

Si el archivo **ubicaciones.txt** no existe o se encuentra en blanco se considera que estamos comenzando una partida nueva. Además, les aseguramos que si existe el archivo materiales.txt todos los valores de los materiales van a ser 0. Es decir:

```
madera 0 0
piedra 0 0
metal 0 0
andycoins 0 0
bombas 0 0
```

Una vez obtenidos los datos de los 3 archivos² se deberá mostrar el siguiente menú:

1. Modificar edificio por nombre.
2. Listar todos los edificios.
3. Mostrar mapa.
4. Comenzar partida.
5. Guardar y salir.

Modificar edificio por nombre

Se deberá verificar que exista el edificio, en caso de que el nombre ingresado sea correcto se le pedirá al usuario que ingrese los nuevos valores de construcción (cantidad de piedra, madera y metal necesarios) para el edificio seleccionado.

² materiales.txt, edificios.txt y mapa.txt

Se le deberá dar la opción al usuario de saltarse un material si no desea modificar ese valor. En caso de desear modificarlo se deberá verificar que el número ingresado sea mayor a 0 y menor a 50000

Por último, si el usuario desea modificar los valores de “obelisco” se le deberá indicar que ese edificio no es modificable.

Listar todos los edificios (igual parte 2)

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 construir³ sin superar el máximo permitido y si me brinda algún tipo de material⁴.

Mostrar mapa (igual parte 2)

Mostrará 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 una segunda matriz con las letras indicadas los tipos de terrenos presentes en el mapa.

Comenzar partida

Se le deberá pedir al usuario que seleccione si desea ser el jugador 1 o el jugador 2. Una vez seleccionado el número de jugador se le pedirá a cada uno que ingrese la coordenada⁵ en la cual desea posicionarse para comenzar el juego. Cuando ambos jugadores se encuentren posicionados se da por comenzado el juego.

Juego

Cada jugador va a contar con puntos de energía, en el primer turno se le asignaran 50 puntos de energía a cada jugador. La energía que no se consuma se guardara para el turno siguiente siempre que no supere los 100 puntos de energía, toda energía por encima de 100 se pierde. **Si el jugador no tiene energía para realizar una acción la misma se cancelará.**

En caso de que se cuente con un archivo ubicaciones.txt que no este en blanco el programa deberá comenzar con el menú de esta sección.

Ahora que el usuario tiene su número de jugador se elegirá de forma aleatoria que jugador tiene el primer turno y se le mostrara el **mapa** junto con el siguiente menú:

1. Construir edificios por nombre.
2. Listar mis edificios construidos.
3. Demoler un edificio por coordenada.
4. Atacar un edificio por coordenada.
5. Reparar un edificio por coordenada.
6. Comprar bombas.
7. Consultar coordenada.
8. Mostrar inventario.
9. Mostrar objetivos.
10. Recolectar recursos producidos.
11. Moverse a una coordenada.

³ Entiéndase por poder construir no superar el máximo de permitidos en esta parte no es necesario verificar los materiales.

⁴ Ver anexo edificios

⁵ Se debe verificar que la misma sea válida, es decir, este dentro del mapa.

12. Finalizar turno.
13. 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 avisará 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 porque no es un casillero construible o porque el mismo se encuentra ocupado por un jugador o edificio.

Energía necesaria: 15

Listar mis edificios construidos

Se deberán listar todos los edificios **construidos por el jugador que lo solicite**, es decir, que haya por lo menos un edificio de este tipo indicando cuantos hay construidos de cada uno, las coordenadas en las que se encuentran y si necesitan o no reparación.

Energía necesaria: 0

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 y **sea del jugador que solicita la demolición** se demolerá y se devolverán la mitad de los materiales utilizados para su construcción.

Energía necesaria: 15

Atacar un edificio por coordenada

Se le pedirá al usuario que ingrese las coordenadas del edificio que desea atacar, en caso de que haya un edificio en esa posición, el usuario tenga una bomba en su inventario y el edificio seleccionado sea del oponente, el mismo se destruirá total o parcialmente siguiendo la siguiente lógica:

- Si es una mina o una fábrica se dañará y podrá ser reparada el próximo turno. Si la misma ya se encuentra dañada, se destruirá totalmente. Es decir, una mina o fábrica requiere de 2 bombas consecutivas (que no se haya reparado luego de la bomba 1) para ser destruida.
- Cualquier otro edificio se destruye totalmente y queda el terreno libre para construir.

Energía necesaria: 30

Reparar un edificio por coordenada

Se le pedirá al usuario que ingrese las coordenadas del edificio que desea reparar, en caso de que haya un edificio en esa posición, sea del usuario que lo solicita y requiera reparación se podrá reparar el mismo si se cuenta con el 25% de cada uno de los materiales que necesitaría para construir uno nuevo.

Energía necesaria: 25

Comprar bombas.

Se le pedirá al usuario que ingrese la cantidad de bombas que desea comprar, cada una costará 100 andycoins. Si no le alcanza se le deberá indicar al usuario que no puede adquirir esa cantidad. Por otro lado, si le alcanza se le deberá indicar la cantidad de bombas que adquirió y el dinero que le queda.

Energía necesaria: 5

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.

Energía necesaria: 0

Mostrar objetivos

Se le mostrara al usuario los objetivos⁶ que tiene que cumplir y el progreso que lleva de cada uno.

Energía necesaria: 0

Mostrar inventario

Se deberá mostrar el inventario del jugador, es decir, los materiales que tiene el mismo.

Energía necesaria: 0

Recolectar recursos producidos

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

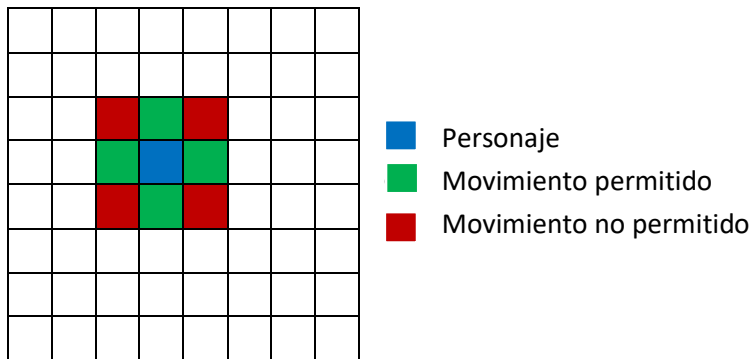
Energía necesaria: 20

Moverse a una coordenada (camino mínimo)

- Para moverse de un lugar a otro, los personajes siempre siguen el camino por el cual gastan la menor cantidad de energía posible esto se explica en detalle en la sección
- **Costos de terreno.**

⁶ Ver sección objetivos

Los personajes pueden moverse a casillas aledañas, pero no pueden desplazarse en diagonal **en un sólo movimiento**, es decir:



Para moverse de un lugar A a un lugar B, el personaje deberá tener la energía suficiente para arribar a destino, en caso contrario, el personaje no se desplazará y el usuario podrá elegir si realizar otra acción o moverse a otro destino.

Para encontrar el camino de menor consumo de energía se deberá utilizar lo visto en clase sobre **grafos**.

Energía necesaria: determinada por el camino.

NOTA: Deberá notarse **claramente** cuál fue el camino seguido por el personaje.

Finalizar turno

Finaliza el turno y deja jugar al otro jugador. Si el jugador tiene 0 puntos de energía el turno se finaliza automáticamente avisándole que se quedó sin energía. Una vez finalizado el turno el jugador recibirá 20 puntos de energía.

Energía necesaria: 0

Lluvia de recursos

Al comienzo de la partida (antes que juegue ningún jugador) y luego de que termine una ronda completa, es decir, haya jugado 1 vez cada jugador, se generarán alrededor del mapa, en los casilleros permitidos, madera, piedra, metal y **andycoins**. Se podrá tener como mucho 1 material por casillero, **sin permitir superposición** y siguiendo la siguiente lógica:

- Se generarán entre 1 y 2 conjuntos de piedras (**cada uno con 100 unidades de piedra**) de forma aleatoria.
- Se generarán entre 0 y 3 conjuntos de maderas (**cada uno con 50 unidades de madera**) de forma aleatoria.
- Se generarán entre 2 y 4 conjuntos de metales (**cada uno con 50 unidades de metal**) de forma aleatoria.

- Se generarán entre 0 y 1 bolsas de andycoins (cada bolsa tendrá 250 andycoins) de forma aleatoria.

Por ejemplo: si debo generar 1 conjunto de piedra significa que buscare 1 posición aleatoria en el mapa en la que posicionare las 100 piedras.

NOTA: La cantidad de conjuntos 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 se puede poner un máximo de iteraciones, la única forma de no posicionar materiales es que no haya **ningún** casillero transitable disponible y en ese caso se le debe avisar al jugador.

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.

Objetivos











El juego se puede ganar de dos maneras cumpliendo el objetivo principal o cumpliendo **2** objetivos secundarios.

Objetivo principal:

-  Más alto que las nubes: construir el obelisco.





Objetivos secundarios:




Al comienzo del juego se le asignaran 3 de estos objetivos a cada jugador de forma aleatoria, si se guarda la partida estos objetivos se pierden y se volverán a asignar otros al volver a jugar.

-  Comprar andypolis: haber juntado 100.000 andycoins a lo largo de la partida (las monedas gastadas también cuentan para este objetivo)
-  Edad de piedra: tener en el inventario 50000 piedras
-  Bombardero: haber usado 5 bombas.
-  Energético: haber terminado un turno con 100 puntos de energía.
-  Letrado: haber construido el máximo posible de escuelas.
-  Minero: haber construido una mina de cada tipo.
-  Cansado: terminar un turno con 0 de energía.
-  Constructor: construir un edificio de cada tipo.
-  Armado: tener 10 bombas en el inventario.
-  Extremista: haber comprado 500 bombas en una partida.

Edificios

Los edificios brindaran los siguientes materiales:

-  Mina: brinda 15 piedras.
-  Aserradero: brinda 25 maderas.
-  Fabrica: brinda 40 metales.
-  Escuela: brinda 25 andycoins.

-  Obelisco: no brinda materiales.
-  Planta eléctrica: brinda 15 de energía.
-  Mina oro: brinda 50 andycoins.

Los edificios generan materiales por turno y se acumulan hasta que el usuario decida recolectarlos.

Mapa

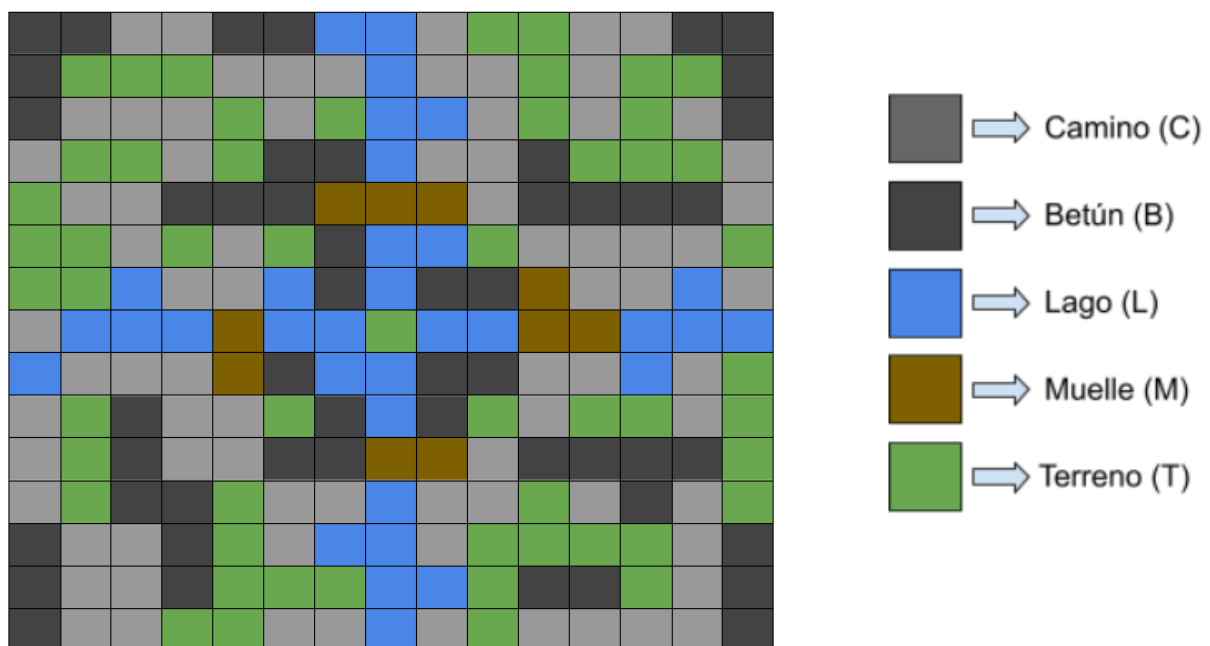
El mapa deberá implementarse con una matriz dinámica y polimórfica.

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

- **Jugador 1: J**
- **Jugador 2: U**
- Mina: M
- **Mina oro: G**
- Aserradero: A
- Fabrica: F
- Escuela: E
- Obelisco: O
- Planta eléctrica: P
- Madera: W
- Piedra: S
- Metal: I
- **Andycoins: C**

El mapa no podrá tener nunca dos materiales, jugadores o edificios en la misma posición. No hay caracteres para bombas ya que nunca están en el mapa.

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, muelles y betún** solo podrán ubicarse **materiales**. Pertenecerán a los casilleros **transitables**.

Costos de terreno

Para la sección de camino mínimo transitar por un terreno tendrá un costo de energía asociado.

Camino

Los caminos son zonas difíciles de transitar y por lo que a todos los jugadores les costará 4 de energía pasar por los mismos.

Betún

El betún es una zona común por lo que a todos los jugadores les costará 0 de energía pasar por los mismos.

Lago

Sabemos que el jugador 2 le tiene fobia al agua, por lo que si quiere pasar por un lago deberá gastar 5 de energía. En cambio, el jugador 1 es un hábil nadador, solo gastará 2 de energía.

Muelle

Al jugador 1 no le gustan los muelles, por lo que si quiere pasar por un muelle deberá gastar 5 de energía. En cambio, el jugador 2 no tiene ningún inconveniente y gastará 2 de energía para pasar por un muelle.

Terreno

Los terrenos son zonas peligrosas siempre hay construcciones y las probabilidades de salir lastimados son altas, si un jugador quiere pasar por un terreno le costará 25 de energía.

Diccionario

Un diccionario es una estructura de datos el cual trabaja con pares de valores, uno es llamado **clave** y el otro **valor**. Los **valores** son accedidos a través de las **claves**.

Para este trabajo práctico, los nombres de los edificios sirven como **claves** y la “receta” del edificio en si como **valor**. Por ejemplo:

```
diccionario.buscar("escuela")
```

Este método nos devuelve la “receta” del edificio cuyo nombre es escuela.

En ese caso, el diccionario deberá ser implementado con un **árbol binario de búsqueda** (ABB).

Aclaraciones

- ✚ El trabajo es de carácter grupal, los grupos estarán compuestos por **4** integrantes. **No están obligados a mantener los grupos del trabajo practico anterior.**
- ✚ Todos los integrantes deberán participar de igual forma en árboles y grafos.

- ✚ Cada grupo deberá decidir cuál de los trabajos prácticos de los integrantes usar como base o si fusionar los mismos. Todos los errores marcados en la corrección del trabajo practico 2 deberán ser corregidos para esta tercera 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.
- ✚ Correcta utilización de objetos.
- ✚ Correcta aplicación de árboles y grafos.

Normas de entrega

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

NombreGrupo_TP3

Por ejemplo:

duo_dinamico_TP3.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 6/12/2021** a las 23.55hs.

Puntaje: 60 puntos.