Proyecto Scrabble Entrega #1

Julián Araque, Pablo González Álvarez, Juan Diego Palacios Toledo
[1]Pontifica Universidad Javeriana, Bogota,
Colombia
j-araque@javeriana.edu.co
P-gonzalez@javeriana.edu.co
ju.palacios@javeriana.edu.co

Abstract. Este documento detalla el desarrollo de un proyecto de Scrable implementado en C++, centrando la atención en la manipulación avanzada de estructuras de datos y aplicación de algoritmos eficientes para gestionar las operaciones fundamentales del juego. A través de la creación de un sistema robusto que permite la inicialización de juegos, la gestión de diccionarios y el calculo de puntaje de palabras, este trabajo ilustra no solo la aplicación práctica de conceptos teóricos de la ingeniería de Sistemas, si no también el proceso de diseño, implementación y prueba de un software complejo. Se enfatiza en la primera entrega del proyecto, presentando el componente de inicialización del juego y el diccionario, así como el sistema de calculo de puntaje, acompañados de un plan de pruebas meticuloso para asegurar la funcionalidad y fiabilidad del sistema. Este proyecto no solo busca proporcionar una solución programática al juego de Scrable, si no también fomentar una reflexión profunda sobre el uso de estructuras de datos complejas y la eficiencia algorítmica, demostrando el impacto de estas en el desarrollo de aplicaciones interactivas y de alto rendimiento.

1. TAD's

TAD Asignacion

- Conjunto mínimo de datos
  - nombreArchivo,(nombreArchivo), string, representa el nombre del archivo a inicializa
  - DiccionarioInicializado, (diccionarioInicializado), bool, indica si el diccionario está inicializado
  - DiccionarioInversoInicializado, (diccionarioInversoInicializado),bool, indica si el diccionario inverso está inicializado
  - palabra, (palabra) ,string, representa una palabra

- Comportamiento

inicializarJuego(nombreArchivo) : void, inicializa el juego con el archivo dado

Postcondiciones: El juego se inicializa con los datos del archivo proporcionado

inicializarInverso(nombreArchivo) : void, inicializa el diccionario inverso con el archivo dado

Postcondiciones: El diccionario inverso se inicializa con los datos del archivo proporcionado

puntajePalabra(palabra) : void, calcula el puntaje de una palabra

Postcondiciones: El puntaje de la palabra se calcula y se realiza alguna acción correspondiente

TAD Scrabble

- Conjunto mínimo de datos

LimpiarPantalla,(limpiarPantalla), void, limpia la pantalla de la consola

Postcondiciones: La pantalla de la consola se limpia

PantallaPrincipal,(pantallaPrincipal), void, muestra la pantalla principal del juego

Postcondiciones: La pantalla principal del juego se muestra en la consola

PausarPantalla,(pausarPantalla), void, pausa la pantalla

Postcondiciones: La pantalla se pausa y espera una acción del usuario

mostrarAyuda(mostrarAyuda), void, muestra información de ayuda sobre el juego

Postcondiciones: Se muestra información de ayuda sobre el juego en la consola

2. Componente 1

El componente 1 se caracteriza por albergar los comandos principales para la inicialización del juego de scrabble, Tales como inicializarJuego, iniciarInverso, puntaje y salir.

Comando y entrada: inicializarJuego o i diccionario.txt
Salidas en pantalla:

```
(Diccionario ya inicializado) El diccionario ya ha sido inicializado.

(Archivo no existe) EL archivo " << nombreArchivo << " no Existe o no puede ser leído.

(Resultado exitoso) El diccionario se ha inicializado correctamente.
```

Descripción: Inicializa el juego mediante el archivo del diccionario y empieza la partida.

```
void inicializarJuego(const std::string& nombreArchivo) {

    if (diccionarioInicializado) {
        std::cout << "(Diccionario ya inicializado) El diccionario ya ha sido
inicializado." << std::endl;
        return;
    }
    std::ifstream archivo(nombreArchivo);
    if (!archivo) {
        std::cerr << "(Archivo no existe) EL archivo " << nombreArchivo << "
no Existe o no puede ser leido" << std::endl;
        return;
    }
    std::string palabra;
    while (archivo >> palabra) {
        bool palabraValida = true;
        for (char c : palabra) {
            if (!std::isalpha(c)) {
                palabraValida = false;
                break;
            }
        }
        if (palabraValida) {
            palabrasValidas.insert(palabra);
        }
    }
    if (palabrasValidas.empty()) {
        std::cout << "(Diccionario ya inicializado) El diccionario ya ha sido
inicializado." << std::endl;
    } else {
        std::cout << "(Resultado exitoso) El diccionario se ha inicializado
correctamente." << std::endl;
    }
    diccionarioInicializado = true;
}
```

Figura No.1 ” Demostración en código función comando inicializarJuego o i”

Comando y entrada: iniciar_inverso o ii diccionario.txt
Salidas:

```
(Diccionario inverso ya inicializado) El diccionario inverso ya ha sido inicializado.

(Archivo no existe) EL archivo " << nombreArchivo << " no Existe o no puede ser leido

(Diccionario inverso ya inicializado) El diccionario inverso ya ha sido inicializado.

(Resultado exitoso) El diccionario inverso se ha inicializado correctamente
```

Descripción: Se inicia el diccionario de forma inversa en donde mediante el archivo .txt se recorre de atrás hacia adelante.

```
void inicializarInverso(const std::string& nombreArchivo) {

        if (diccionarioInversoInicializado) {
            std::cout << "(Diccionario inverso ya inicializado) El diccionario
inverso ya ha sido inicializado." << std::endl;
            return;
        }
        std::ifstream archivo(nombreArchivo);
        if (!archivo) {
            std::cerr << "(Archivo no existe) EL archivo " << nombreArchivo << "
no Existe o no puede ser leido" << std::endl;
            return;
        }
        std::string palabra;
        while (archivo >> palabra) {
            bool palabraValida = true;
            for (char c : palabra) {
                if (!std::isalpha(c)) {
                    palabraValida = false;
                    break;
                }
            }
            if (palabraValida) {
                std::string palabraInversa(palabra.rbegin(), palabra.rend());
                palabrasValidasInverso.insert(palabra);
            }
        }
        if (palabrasValidasInverso.empty()) {
            std::cout << "(Diccionario inverso ya inicializado) El diccionario
inverso ya ha sido inicializado." << std::endl;
        } else {
            std::cout << "(Resultado exitoso) El diccionario inverso se ha
inicializado correctamente." << std::endl;
        }
        diccionarioInversoInicializado = true;
    }
```

Figura No. 2” Demostración en código función comando inicializarInverso o ii”

Comando y Entrada: puntajePalabra “palabra”

Salidas:

`(Resultado exitoso) La palabra tiene un puntaje de` *`puntaje`*

`(Error) La palabra contiene letras inválidas.`

`(Palabra no existe) La palabra no existe en el diccionario.`

Descripción: Se obtiene el puntaje de cierta palabra mediante el diccionario y si no se encuentra se remite el mensaje.

```
void puntajePalabra(const std::string& palabra) {
    const std::unordered_set<std::string>& diccionario = obtenerDiccionario();
    const std::unordered_set<std::string>& diccionarioInverso =
obtenerDiccionarioInverso();
    // verificar si la palabra existe en el diccionario
    if (diccionario.find(palabra) != diccionario.end() ||
diccionarioInverso.find(palabra) != diccionarioInverso.end()) {
        // Verificar si la palabra contiene solo letras válidas
        bool palabraValida = true;
        for (char c : palabra) {
            if (!std::isalpha(c)) {
                palabraValida = false;
                break;
            }
        }
        if (palabraValida) {
            int puntaje = calcularPuntajePalabra(palabra);
            if (puntaje != -1) {
                std::cout << "(Resultado exitoso) La palabra tiene un puntaje
de " << puntaje << "." << std::endl;
                return;
            } else {
                std::cerr << "(Error) La palabra contiene letras inválidas." <<
std::endl;
                return;
            }
        } else {
            std::cerr << "(Error) La palabra contiene letras inválidas." <<
std::endl;
            return;
        }
    } else {
        std::cerr << "(Palabra no existe) La palabra no existe en el
diccionario." << std::endl;
        return;
    }
}
/
```

Figura No.3 “Demostración en código función comando puntajePalabra o pp”

Comando: -
Salidas: Retorna el puntaje
Descripción: Funcion que permite calcular el puntaje de la palabra por ende solo tiene la salida del puntaje de cada palabra.

```
int calcularPuntajePalabra(const std::string& palabra) {
    //Tabla de puntajes por letra
    std::unordered_map<char, int> puntajes = {
        {'E', 1}, {'A', 1}, {'I', 1}, {'O', 1}, {'N', 1}, {'R', 1}, {'T',
1}, {'L', 1}, {'S', 1}, {'U', 1},
        {'D', 2}, {'G', 2},
        {'B', 3}, {'C', 3}, {'M', 3}, {'P', 3},
        {'F', 4}, {'H', 4}, {'V', 4}, {'W', 4}, {'Y', 4},
        {'K', 5},
        {'J', 8}, {'X', 8},
        {'Q', 10}, {'Z', 10}
    };
    int puntaje = 0;
    for (char c : palabra) {
        // convertir a mayusculas para buscar en la tabla de puntajes
        char letra = std::toupper(c);
        if (puntajes.find(letra) != puntajes.end()) {
            puntaje += puntajes[letra];
        } else  {
            // La letra no esta en la tabla de puntajes
            return -1;
        }
    }
    return puntaje;
}
```

Comando: salir
Salidas: Ninguna

```
            case 's':
                if (comando == "salir" || comando == "s") {
                    limpiarPantalla();
                    cout << "este es el comando salir" << endl;
                    cout << "saliendo del juego" << endl;
                    juego_terminado = true;
                    pausarPantalla();
                }
```

Figura No.4 “Demostración en código comando salir o s”
Descripción: Termina la ejecución del juego y sale. Utilizamos parte del switch para poder utilizar el comando salir o s.

3. **Condiciones para el procedimiento principal**

1. Que exista un archivo diccionario donde se aloje el diccionario con las letras para el juego.
2.

4. **Operaciones auxiliares (comandos)**

A continuación, se presentan los comandos adicionales, que aún se encuentran en desarrollo para las siguientes entregas, estos comandos se utilizaran en los siguientes componentes y completaran el juego.

- ia - iniciarArbol (alias: ia)
- iai - iniciarArbolInv (alias: iai)
- pt - pasarTurno (alias: pt)
- ppr - palabraPrefijo (alias: ppr)
- psu - palabraSufijo (alias: psu)
- ppa - posiblePalabra (alias: ppa)
- gp - grafoPalabra (alias: gp)

5. **Plan de Pruebas**

Comando ‘InicializarJuego’ (o ‘i’)

Objetivo

Verificar que el comando ‘InicializarJuego’ inicializa correctamente el sistema con el archivo de diccionario proporcionado, maneja errores de manera apropiada y evita la realización cuando ya ha sido ejecutado

Precondiciones

- El sistema debe estar en un estado no inicializado al comenzar cada prueba.
- Deben existir archivos de prueba validos e inválidos para simular diferentes escenarios.

Casos de Prueba

1. Archivo valido
   Descripción: Inicializar el juego con un archivo de diccionario valido
   - Entrada: 'i'

   esperar al programa que pida la entrada del archivo

   - Entrada: 'englishWords-1.txt' (asegurando de que 'englishWords-1.txt' es un archivo existente y valido)

   Resultado Esperado: Mensaje de éxito indicando que el diccionario se ha inicializado correctamente
2. Archivo no Existe
   Descripción: Intentar inicializar el juego con un archivo de diccionario que no existe.
   - Entrada: 'i'

   esperar al programa que pida la entrada del archivo

   - Entrada: 'archivoInexistente.txt'

   Resultado Esperado: Mensaje de error indicando que el archivo no existe o no puede ser leído
3. Formato de Archivo Incorrecto

   Descripción: Usar un archivo con formato incorrecto (por ejemplo, con símbolos inválidos en las palabras).

   - Entrada: 'i'

   esperar al programa que pida la entrada del archivo

   - Entrada: 'diccionarioPalabrasInvalidas.txt' (se entiende como carácter invalido a toda palabra que tenga acento o caracteres en ella (Ej: camarón!) en este caso la palabra cuenta con dos caracteres inválidos (ó y !) esto siendo palabras invalidas)

   Resultado Esperado: Mensaje de error indicando un problema con el formato del archivo.

4. Re-inicialización

   Descripción: Intentar re-inicializar el juego después de haber sido inicializado correctamente,

   - Entrada: 'i'

   esperar al programa que pida la entrada al archivo

   - Entrada: '<texto cualquier>'

   Resultado Esperado: Mensaje de error o advertencia indicando que el diccionario ya ha sido inicializado.

5. Comando incompleto

   Descripción: Intentar inicializar sin introducir nada después que se pide que se ingrese un archivo.

   - Entrada: ‘i’

   esperar al programa que pida la entrada al archivo

   - Entrada: NA (Referencia de no escribir ningún texto)

   Resultado Esperado: Mensaje de error indicando que no se ingresó ningún archivo.

Postcondiciones

- Para el caso de éxito, el sistema debe estar en un estado inicializado, listo para recibir otros comandos relacionados con el juego
- Para los casos de erros, el sistema debe permanece en un estado no inicializado sin afectar la integridad del juego.

Criterios de Aceptación

- El comando debe manejar adecuadamente todos los escenarios previstos, mostrando mensajes claros y precisos que guíen al usuario en caso de errores
- El sistema no debe permitir la re-inicializacion una vez que ha sido inicializado correctamente, a menos que se reinicie completamente el programa.

Comando ‘PuntajePalabra’ (o ‘pp’)

Objetivo

Verificar que el comando ‘puntajePalabra’ inicializa correctamente el sistema con el archivo de diccionario proporcionado para detectar el puntaje de la palabra ingresadaPrecondiciones

- El sistema debe tener un diccionario ya ingresado

Casos de Prueba

1. Palabra valida
   Descripción: Inicializar el juego con un archivo de diccionario valido
   - Entrada: ‘pp’

   esperar al programa que pida la entrada del archivo

   - Entrada: ‘Alguna palabra’ (asegurando de que la palabra este en ‘englishWords-1.txt’)

   Resultado Esperado: Mensaje de éxito indicando el puntaje de la palabra

2. Palabra no Existe

- Descripción: Ingresar una palabra que no esté en el diccionarioEntrada: ‘i’

esperar al programa que pida la entrada del archivo

- Entrada: Arbol
- Resultado Esperado: ‘La palabra no existe en el diccionario.’

6. Resultado plan de pruebas

Ingresar comando (‘i o inicializarJuego’):

```
este es el comando inicializarJuego
Comando en progreso...
Ingrese el nombre del archivo:
```

Ingresar un diccionario:

```
este es el comando inicializarJuego
Comando en progreso...
Ingrese el nombre del archivo: englishWords-1.txt
(Diccionario ya inicializado) El diccionario ya ha sido inicializado.
saliendo del comando inicializarJuego
$
```

Ingresar un diccionario que no existe:

```
este es el comando inicializarJuego
Comando en progreso...
Ingrese el nombre del archivo: diccionario
(Archivo no existe) EL archivo diccionario no Existe o no puede ser leido
saliendo del comando inicializarJuego
$
```

Ingresar comando (inicializarInverso o ii) e ingresar el nombre del archivo

```
este es el comando inicializarInverso
Comando en progreso...
Ingrese el nombre del archivo: englishWords-1.txt
(Diccionario inverso ya inicializado) El diccionario inverso ya ha sido inicializado.
saliendo del comando inicializarInverso
$
```

Ingresar comando (puntajePalabra o pp) para ver el puntaje de una palabra del diccionario

```
este es el comando puntajePalabra
Comando en progreso...
Ingrese la palabra para calcular su puntaje: Yoknapatawpha
(Resultado exitoso) La palabra tiene un puntaje de 30.
saliendo del comando puntajePalabra
$
```

Ingresar comando (puntajePalabra o pp) para ver el puntaje de una palabra del diccionario (La palabra no existe)

```
este es el comando puntajePalabra
Comando en progreso...
Ingrese la palabra para calcular su puntaje: arbol
(Palabra no existe) La palabra no existe en el diccionario.
saliendo del comando puntajePalabra
$
```

7. Conclusiones

En desarrollo....