Expresiones Regulares en C++ con la Librería `regex`

Las expresiones regulares (regex) son una poderosa herramienta para la manipulación y búsqueda de cadenas. En C++, la librería `<regex>` proporciona un soporte robusto para trabajar con expresiones regulares.

1. Búsqueda de patrones con `std::regex_search`

Esta función se utiliza para buscar un patrón dentro de una cadena. Devuelve `true` si el patrón se encuentra.

Ejemplo 1: Buscar un número en una cadena

```cpp
#include <iostream>
#include <regex>

int main() {
    std::string texto = "El año es 2024";
    std::regex numero("\d+");
    if (std::regex_search(texto, numero)) {
        std::cout << "Se encontró un número en el texto." << std::endl;
    }
}
```

Salida esperada: `Se encontró un número en el texto.`

Ejemplo 2: Buscar una palabra específica

```cpp
#include <iostream>
#include <regex>

int main() {
    std::string texto = "Bienvenido a la programación en C++";
    std::regex palabra("programación");
    if (std::regex_search(texto, palabra)) {
        std::cout << "Se encontró la palabra 'programación' en el texto." <<
std::endl;
    }
}
```

Salida esperada: `Se encontró la palabra 'programación' en el texto.`

Ejemplo 3: Buscar un correo electrónico

```cpp
#include <iostream>
#include <regex>

int main() {
    std::string texto = "Mi correo es ejemplo@correo.com";
```

```
    std::regex correo("\w+@\w+\.\w+");
    if (std::regex_search(texto, correo)) {
        std::cout << "Se encontró un correo electrónico en el texto." << std::endl;
    }
}
```

Salida esperada: `Se encontró un correo electrónico en el texto.`

2. Reemplazo de patrones con `std::regex_replace`

Esta función se utiliza para reemplazar todas las ocurrencias de un patrón en una cadena por otro texto.

Ejemplo 1: Reemplazar espacios por guiones

```
#include <iostream>
#include <regex>

int main() {
    std::string texto = "Hola Mundo";
    std::regex espacio("\s+");
    std::string resultado = std::regex_replace(texto, espacio, "-");
    std::cout << "Texto después de reemplazo: " << resultado << std::endl;
}
```

Salida esperada: `Texto después de reemplazo: Hola-Mundo`

Ejemplo 2: Reemplazar números por asteriscos

```
#include <iostream>
#include <regex>

int main() {
    std::string texto = "Mi número es 123456";
    std::regex numeros("\d");
    std::string resultado = std::regex_replace(texto, numeros, "*");
    std::cout << "Texto después de reemplazo: " << resultado << std::endl;
}
```

Salida esperada: `Texto después de reemplazo: Mi número es ******`

Ejemplo 3: Reemplazar un dominio de correo electrónico

```
#include <iostream>
#include <regex>

int main() {
    std::string texto = "Contactame en ejemplo@correo.com";
    std::regex dominio("@\w+\.\w+");
    std::string resultado = std::regex_replace(texto, dominio, "@dominio.com");
    std::cout << "Texto después de reemplazo: " << resultado << std::endl;
}
```

Salida esperada: `Texto después de reemplazo: Contactame en ejemplo@dominio.com`

3. División de cadenas con `std::regex_token_iterator`

Esta función permite dividir una cadena en partes utilizando un patrón como delimitador.

Ejemplo 1: Dividir por espacios

```cpp
#include <iostream>
#include <regex>

int main() {
    std::string texto = "Dividir esta cadena por espacios";
    std::regex espacio("\s+");
    std::sregex_token_iterator iter(texto.begin(), texto.end(), espacio, -1);
    std::sregex_token_iterator end;

    while (iter != end) {
        std::cout << *iter++ << std::endl;
    }
}
```

Salida esperada:

```
Dividir
esta
cadena
por
espacios
```

Ejemplo 2: Dividir una lista de correos electrónicos

```cpp
#include <iostream>
#include <regex>

int main() {
    std::string texto = "correo1@dominio.com,correo2@dominio.com,correo3@dominio.com";
    std::regex coma(",");
    std::sregex_token_iterator iter(texto.begin(), texto.end(), coma, -1);
    std::sregex_token_iterator end;

    while (iter != end) {
        std::cout << *iter++ << std::endl;
    }
}
```

Salida esperada:

```
correo1@dominio.com
correo2@dominio.com
correo3@dominio.com
```

Ejemplo 3: Dividir una URL en componentes

```cpp
#include <iostream>
#include <regex>

int main() {
    std::string url = "https://www.ejemplo.com/path/recurso?query=param";
    std::regex delimitador("/|\?|=");
    std::sregex_token_iterator iter(url.begin(), url.end(), delimitador, -1);
    std::sregex_token_iterator end;

    while (iter != end) {
        std::cout << *iter++ << std::endl;
    }
}
```

Salida esperada:

```
https:
www.ejemplo.com
path
recurso
query
param
```