You cannot use std::stringstream with the extraction operator >> to store whitespace, because this operator is designed to skip all leading whitespace and use any subsequent whitespace as a delimiter.
To store a string including its whitespace, you should use std::getline with std::stringstream. std::getline can extract a substring from a stream up to a specified delimiter, and by using a character that is not present in your data, or by manually parsing the string, you can preserve all whitespace.

Method 1: Reading Character by Character

This approach gives you the most control. You can iterate through the string and build up "word" and "whitespace" components manually.

C++

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>

int main() {
    std::string line = "Hello   world!  This is a test.";
    std::stringstream ss(line);

    std::vector<std::string> parts;
    std::string current_part;
    char ch;

    while (ss.get(ch)) {
        if (isspace(ch)) {
            if (!current_part.empty() && !isspace(current_part.back())) {
                parts.push_back(current_part); // Store the word
                current_part.clear();
            }
        } else {
            if (!current_part.empty() && isspace(current_part.back())) {
                parts.push_back(current_part); // Store the whitespace
                current_part.clear();
            }
        }
        current_part += ch;
```

```
    }
    if (!current_part.empty()) {
        parts.push_back(current_part);
    }

    for (const auto& part : parts) {
        std::cout << "Part: [" << part << "]" << std::endl;
    }

    return 0;
}
```

This method is complex because it requires you to manage state (what you've just read) to decide when to store a complete word or whitespace block.

Method 2: The Best Approach with find_first_of

A much more robust and standard way to solve this problem is to abandon stringstream for the splitting task and use the built-in string functions find_first_of and find_first_not_of. This gives you precise control over splitting the string into words and whitespace blocks.
This method works by manually finding the start and end of each sequence of non-space characters (words) and then the start and end of each sequence of space characters (whitespace).

C++

```
#include <iostream>
#include <string>
#include <vector>

int main() {
    std::string line = "Hello   world!  This is a test.";
    std::vector<std::string> parts;

    std::string::size_type pos = 0;

    while (pos < line.size()) {
        // Find the start of the next word (first non-whitespace character)
        std::string::size_type word_start = line.find_first_not_of(" \t\n\r", pos);
        if (word_start == std::string::npos) {
            break; // No more words
```

```
        }

        // Find the end of the word
        std::string::size_type word_end = line.find_first_of(" \t\n\r", word_start);

        // Extract the word and add it to the vector
        parts.push_back(line.substr(word_start, word_end - word_start));

        // Find the start of the next whitespace block
        std::string::size_type space_start = word_end;
        if (space_start == std::string::npos) {
            break; // End of string after the word
        }

        // Find the end of the whitespace block
        std::string::size_type space_end = line.find_first_not_of(" \t\n\r", space_start);

        // Extract the whitespace block and add it to the vector
        parts.push_back(line.substr(space_start, space_end - space_start));

        // Update position for the next iteration
        pos = space_end;
    }

    // Print the parts to verify
    for (const auto& part : parts) {
        std::cout << "Part: [" << part << "]" << std::endl;
    }

    return 0;
}
```

This code snippet is the correct way to handle your problem. The output will show each word and each block of whitespace as separate strings, exactly as they appear in the original line.