LSP: Task -1

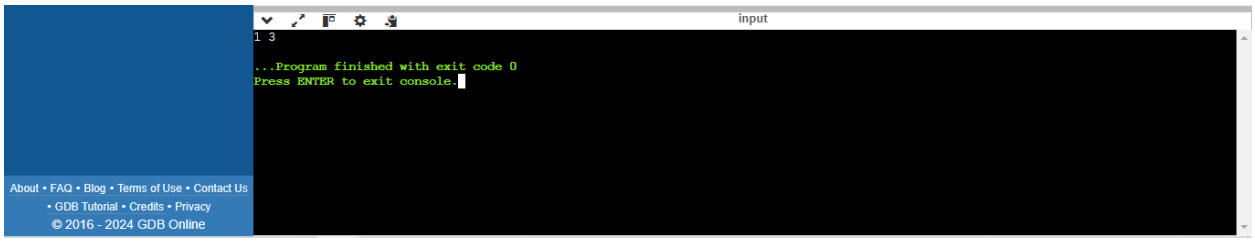
1. Implement a custom dynamic array class that supports basic operations like insertion, deletion, resizing, and clearing.

```
/* #include <iostream>

#include <stdexcept>


template <typename T>

class DynamicArray {

private:

    T* data;

    size_t capacity;

    size_t size;


    void resize(size_t new_capacity) {

        T* new_data = new T[new_capacity];

        for (size_t i = 0; i < size; ++i) {

            new_data[i] = data[i];

        }

        delete[] data;

        data = new_data;

        capacity = new_capacity;

    }


public:

    DynamicArray() : data(nullptr), capacity(0), size(0) {}
```

```
void insert(const T& value) {
    if (size == capacity) {
        resize(capacity == 0 ? 1 : capacity * 2);
    }
    data[size++] = value;
}

void remove(size_t index) {
    if (index >= size) {
        throw std::out_of_range("Index out of range");
    }
    for (size_t i = index; i < size - 1; ++i) {
        data[i] = data[i + 1];
    }
    --size;
}

void clear() {
    delete[] data;
    data = nullptr;
    capacity = 0;
    size = 0;
}

T& operator[](size_t index) {
    if (index >= size) {
```

```
            throw std::out_of_range("Index out of range");
        }
        return data[index];
    }

    size_t getSize() const {
        return size;
    }

    ~DynamicArray() {
        delete[] data;
    }
};

int main() {
    DynamicArray<int> arr;
    arr.insert(1);
    arr.insert(2);
    arr.insert(3);
    arr.remove(1);

    for (size_t i = 0; i < arr.getSize(); ++i) {
        std::cout << arr[i] << " ";
    }

    arr.clear();
    return 0;
```

```
} */
```

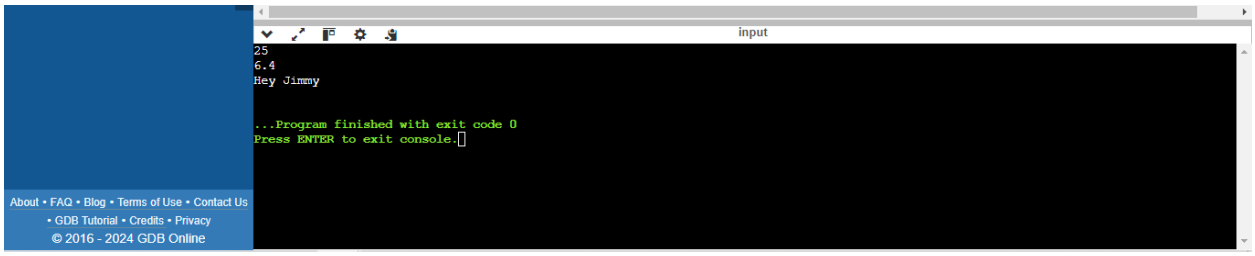


//2. Create a template-based stack class supporting push, pop, and peek operations. Implement it for different data types like int, float, and std::string.

```
/*#include <iostream>
#include <vector>
#include <stdexcept>

template <typename T>
class Stack {
private:
    std::vector<T> elements;

public:
    void push(const T& value) {
        elements.push_back(value);
    }

    void pop() {
        if (elements.empty()) {
```

```
            throw std::out_of_range("Stack is empty");
        }
        elements.pop_back();
    }

    T& peek() {
        if (elements.empty()) {
            throw std::out_of_range("Stack is empty");
        }
        return elements.back();
    }

};

int main() {
    Stack<int> intStack;
    Stack<float> floatStack;
    Stack<std::string> stringStack;

    intStack.push(34);
    floatStack.push(6.4f);
    stringStack.push("Hey Jimmy");

    std::cout << intStack.peek() << std::endl;
    std::cout << floatStack.peek() << std::endl;
    std::cout << stringStack.peek() << std::endl;
```

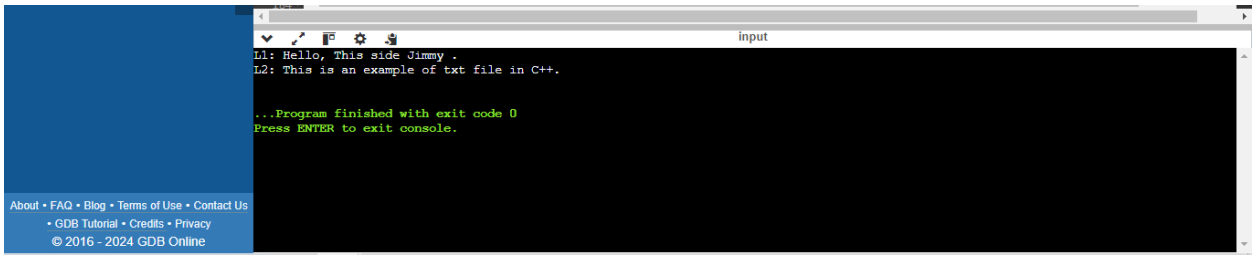
```
    return 0;
} */
```



//3. Write a program that reads from a file and handles various exceptions such as file not found, read errors, and unexpected data formats.

```
/* #include <iostream>
#include <fstream>
#include <stdexcept>

int main() {

    std::ofstream outFile("jim.txt");
    if (outFile.is_open()) {
        outFile << "L1: Hello, This side Jimmy .\n";
        outFile << "L2: This is an example of txt file in C++.\n";
        outFile.close();
    } else {
        std::cerr << "Error: Unable to create file" << std::endl;
        return 1;
    }
```

```
    // Read from the file
    std::ifstream file("jim.txt");

    try {
        if (!file.is_open()) {
            throw std::runtime_error("File not found");
        }

        std::string line;
        while (std::getline(file, line)) {
            if (line.empty()) {
                throw std::runtime_error("Unexpected data format");
            }
            std::cout << line << std::endl;
        }

        file.close();
    } catch (const std::exception& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}*/
```


input
L1: Hello, This side Jimmy .
L2: This is an example of txt file in C++.
...Program finished with exit code 0
Press ENTER to exit console.
About • FAQ • Blog • Terms of Use • Contact Us
• GDB Tutorial • Credits • Privacy
© 2016 - 2024 GDB Online