Lab 08
Tách chuỗi, tạo đối tượng

Lập trình hướng đối tượng

Mục tiêu	1. Tách một chuỗi thành các chuỗi con 2. Chuyển đổi từ chuỗi thành đối tượng

1 Hướng dẫn khởi đầu

Mô tả bài tập

Cho trước mảng các số nguyên được lưu trong chuỗi như sau:

"41, 817, 12, 9371, 154"

Hãy tách chuỗi này ra thành mảng các chuỗi con "41", "817", "12", "9371", "154".

Sau đó khởi tạo mảng các số nguyên bằng cách chuyển mỗi chuỗi con thành số để có mảng:

[41, 817, 12, 9371, 154]

Hướng dẫn cài đặt

Bước 1: Tạo mới dự án

- Chọn loại dự án là **C++ / Console Application**.

- Đặt tên solution là: **Tokenizer**. Đặt tên project là **IntArrayTokenizer**

- Nếu sử dụng Visual Studio 2017 trở lên cần vô hiệu hóa **Precompiled header** bằng cách nhấn phải vào project chọn Properties. Vào mục **C / C++** > **All Options**, tìm tới tùy chọn **Precompiled header** và chọn **Not using precompiled headers**.

Bước 2: Viết hàm main để tách chuỗi như sau

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int main() {
    vector<string> tokens;

    string line = "41, 817, 12, 9371, 154";
    string seperator = ", "; // Kí tự phân cách

    int startPos = 0; // Vị trí bắt đầu tìm
    size_t foundPos = line.find(seperator, startPos);

    while (foundPos != string::npos) { // Vẫn còn tìm thấy
        int count = foundPos - startPos;
        string token = line.substr(startPos, count);
        tokens.push_back(token);

        // Cập nhật vị trí bắt đầu tìm lại
        startPos = foundPos + seperator.length();

        // Tiếp tục tìm
        foundPos = line.find(seperator, startPos);
    }

    // Phần còn sót lại chính là phần tử cuối
    int count = line.length() - startPos;
    string token = line.substr(startPos, count);
    tokens.push_back(token);

    // Lần lượt chuyển các token thành số
    vector<int> numbers;

    for (auto i = 0; i < tokens.size(); i++) {
        auto num = stoi(tokens[i]);
        numbers.push_back(num);
    }

    // Duyệt qua các số và in ra màn hình
    for (auto i = 0; i < numbers.size(); i++) {
        cout << numbers[i] << " ";
    }
}
```

Chạy chương trình và thấy mảng đã được in ra theo dạng từng số nguyên.

Bước 3: Cải tiến trở thành lớp Tokenizer như sau

```
class Tokenizer {
public:
    static vector<string> Parse(string line, string seperator) {
        vector<string> tokens;

        int startPos = 0; // Vị trí bắt đầu tìm
        size_t foundPos = line.find(seperator, startPos);

        while (foundPos != string::npos) { // Vẫn còn tìm thấy
            int count = foundPos - startPos;
            string token = line.substr(startPos, count);
            tokens.push_back(token);

            // Cập nhật vị trí bắt đầu tìm lại
            startPos = foundPos + seperator.length();

            // Tiếp tục tìm
            foundPos = line.find(seperator, startPos);
        }

        // Phần còn sót lại chính là phần tử cuối
        int count = line.length() - startPos;
        string token = line.substr(startPos, count);
        tokens.push_back(token);

        return tokens;
    }
};
```

```
int main() {
    string line = "41, 817, 12, 9371, 154";
    string seperator = ", "; // Kí tự phân cách
    vector<string> tokens = Tokenizer::Parse(line, seperator);

    // Lần lượt chuyển các token thành số
    vector<int> numbers;

    for (auto i = 0; i < tokens.size(); i++) {
        auto num = stoi(tokens[i]);
        numbers.push_back(num);
    }

    // Duyệt qua các số và in ra màn hình
    for (auto i = 0; i < numbers.size(); i++) {
        cout << numbers[i] << " ";
    }
}
```

2 Bài tập vận dụng

Yêu cầu

1. Thực hiện định nghĩa lớp theo thiết kế cho trước vào tập tin .h.
2. Thực hiện cài đặt lớp trong tập tin .cpp cho lớp tương ứng.

   + Cài đặt hàm tĩnh Parse(string) trả ra đối tượng từ chuỗi
3. Viết các đoạn mã nguồn kiểm tra việc định nghĩa lớp trong hàm main.

Danh sách các lớp cần cải tiến cụ thể

1. Lớp **Đường thẳng** có hai thành phần **Điểm**: **Bắt đầu** và **Kết thúc**.

   Line* Line::Parse("(3, 4), (5, 9)")
2. Lớp **Hình chữ nhật** có hai thành phần **Điểm**: **Trái trên** và **Phải Dưới**

   Rectangle* Rectangle::Parse("(6, 15), (1, 20)")
3. Lớp **Hình tam giác** có ba thành phần **Điểm** ứng với 3 đỉnh : **a, b, c**.

   Triangle* Triangle::Parse("(6, 15), (1, 20), (61, 92)")
4. Lớp **Hình tròn** có 2 thành phần: **tâm** (Lớp **Điểm**) và **bán kính** (số thực).

   Circle* Circle::Parse("(4, 3), 1.8")
5. Lớp **Phân số** có 2 thành phần: **tử** (số nguyên) và **mẫu** (số nguyên)

   Fraction* Fraction::Parse("6/12")
6. Lớp **Sinh viên** có 3 thành phần: **họ** (chuỗi), **tên lót** (chuỗi) và **tên** (chuỗi).

   Student* Student::Parse("Nguyen Viet Long")
7. Lớp **Mảng động** (**DynamicArray**)

   DynamicArray* DynamicArray::Parse("5, 8, 12, 15, 612, 19")
8. Lớp **Thời gian** (**Time**) với 3 thành phần **ngày** (số nguyên), **tháng** (số nguyên), **năm** (số nguyên)

   Time* Time::Parse("12:13:46")
9. Lớp **Ngày Tháng** (**Date**) với 3 thành phần **giờ** (số nguyên), **phút** (số nguyên), **giây** (số nguyên)

   Date* Date::Parse("24/12/2018")

3 Bài tập vận dụng khó siêu cấp vô địch

Yêu cầu

Hàm Parse giả định luôn luôn tốt đẹp, không bắt và xử lí ngoại lệ.

Hãy cài đặt hàm tĩnh **TryParse**, hàm này sẽ trả ra đối tượng null nếu parse không thành công đối tượng vì mọi lí do. Ngược lại nếu thành công sẽ trả ra đối tượng đã được chuyển đổi từ chuỗi sang.

Gợi ý: Với mọi exception bắt được, return null.

Sử dụng regular expression để kiểm tra chuỗi có tuân theo định dạng cho trước không.

Bài tập này chỉ dành cho người thích thử thách.

4 Hướng dẫn nộp bài

Trước khi nộp cần chú ý:

- Lấy tập tin exe được biên dịch sẵn trong thư mục Debug, copy nó ra thư mục Release bên ngoài mã nguồn.

- Xóa hết tất cả các tập tin trung gian trong quá trình biên dịch bằng cách chọn **Build** > **Clean**.

- Chú ý thư mục ẩn **.vs** rất nặng. Cần hiển thị file ẩn mới thấy và xóa nó đi được.

> Nếu bạn muốn biết cách làm đúng thì cần tự tìm cách build ở chế độ Release và copy file exe kết quả ra bên ngoài để nộp mới đúng. Tuy nhiên nếu chưa hiểu ý nghĩa thì cứ lấy đại file exe có sẵn đi nộp cũng được (hiện tại đang trong thư mục Debug ứng với chế độ biên dịch Debug)

Tổ chức bài nộp

+ Thư mục **Source**: chứa mã nguồn đã được clean

+ Thư mục **Release**: chứa tập tin thực thi đã được biên dịch từ mã nguồn

+ Tập tin **readme.txt**: chứa thông tin sinh viên, gồm MSSV và họ tên. Ghi chú kèm các thông tin giáo viên cần chú ý khi chấm bài.

Để nộp bài, nén tất cả lại và đặt tên với định dạng **MSSV.zip** hoặc **MSSV.rar** và nộp.

Nếu làm đúng các bước trên file này sẽ có kích thước < 100 KB!

(Tuy nhiên cũng đừng quá lo lắng nếu nó khác con số trên, miễn < 12 MB để nộp được trên moodle là okie nhé!)

Ngoài lề: Để đảm bảo sau này nhìn vào file nén còn biết ngay nó làm gì, ta nên thêm vào một số thông tin theo sau MSSV. Ví dụ: 0712221-Lab08-Tokenizer.zip. Tuy nhiên việc này là KHÔNG bắt buộc nhé.

-- HẾT --