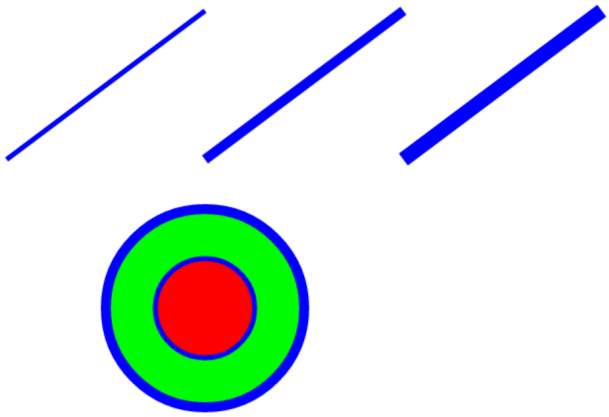
BÁO CÁO THỰC HÀNH



CHỦ ĐỀ: XỬ LÍ TỆP TIN SVG

Tên: Đỗ Thành Nhơn
Lớp: 15CNTN
MSSV: 1512387
GV: Nguyễn Vinh Tiệp – Đỗ Nguyên Kha

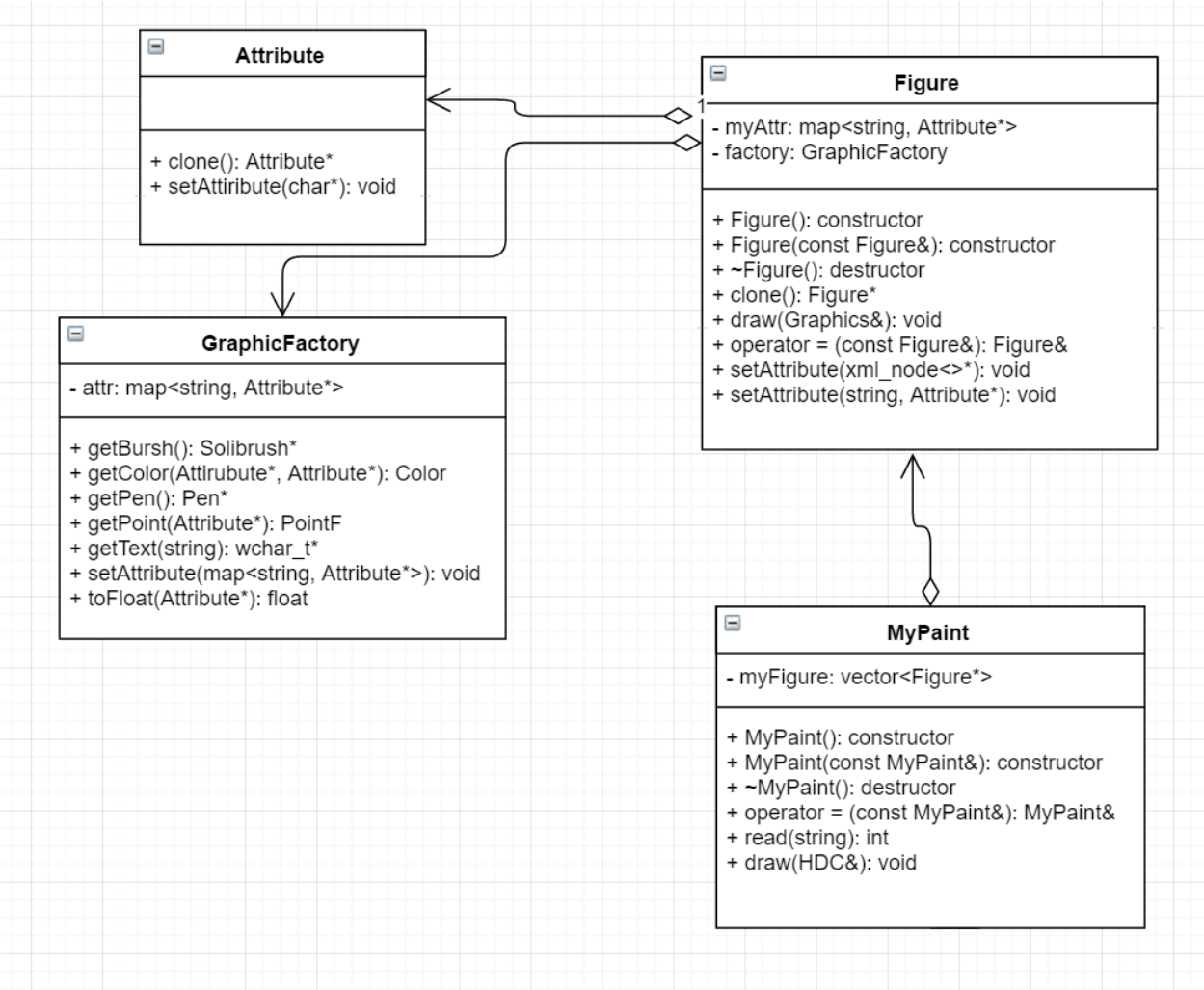
PHỤ LỤC

I. Table of Contents

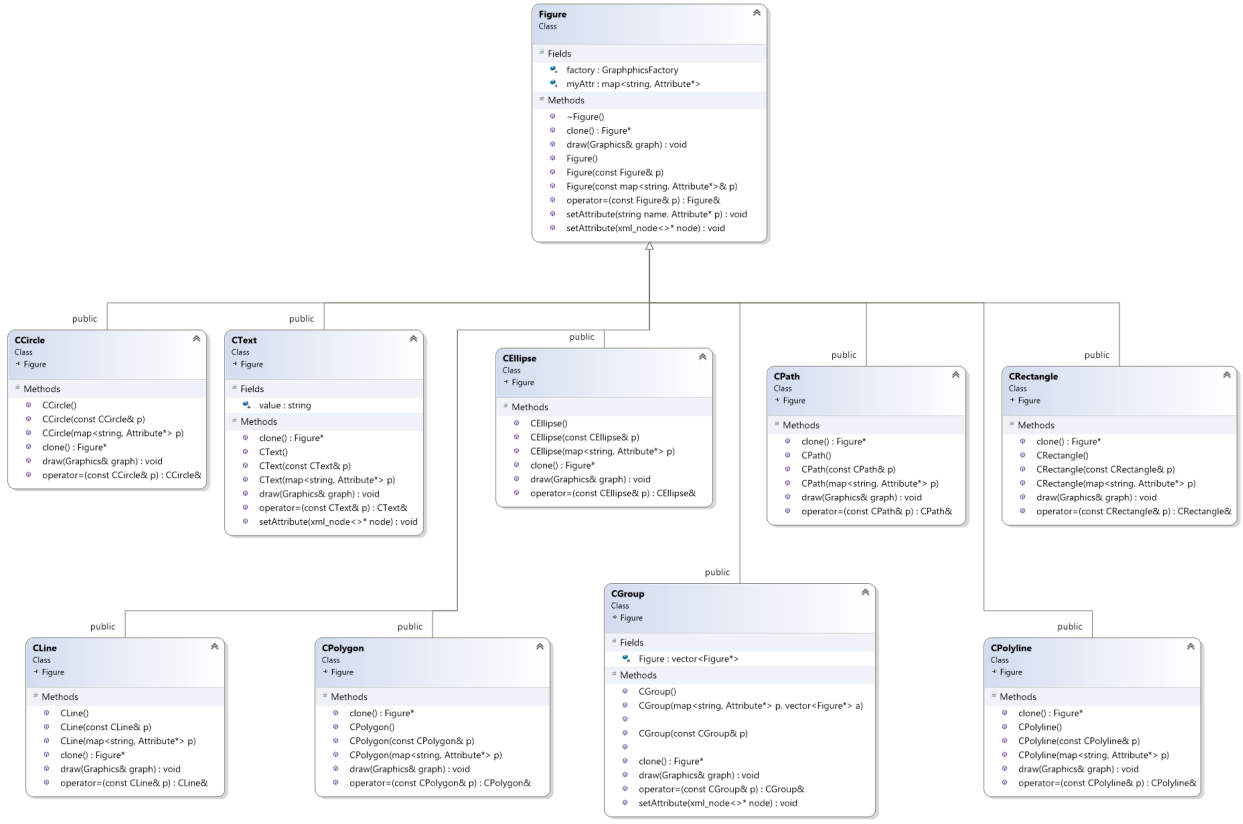
I. Tổng quan đồ án.

1. Đọc tệp tin SVG và xử lí dữ liệu bằng thư viện RapidXML.
2. Thiết kế các lớp đối tượng cho các loại hình và thuộc tính của chúng.
3. Từ dữ liệu đầu vào, khởi tạo các đối tượng hình học tương ứng và trang bị các thuộc tính hình học của chúng.
4. Sử dụng thư viện đồ hoạ để render ra màn hình và so sánh với ảnh gốc (Thư viện GDI+ trên Win32 API).

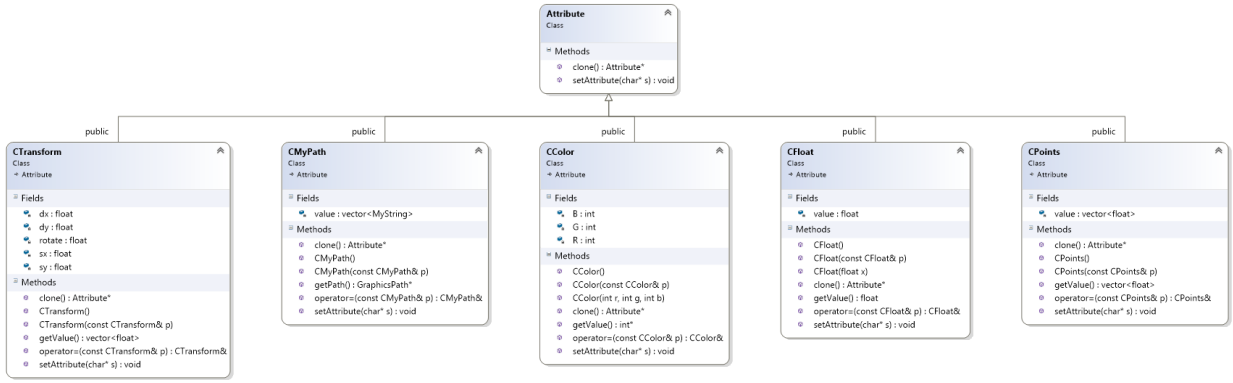
II. Cách phân chia và tổ chức các lớp đối tượng.



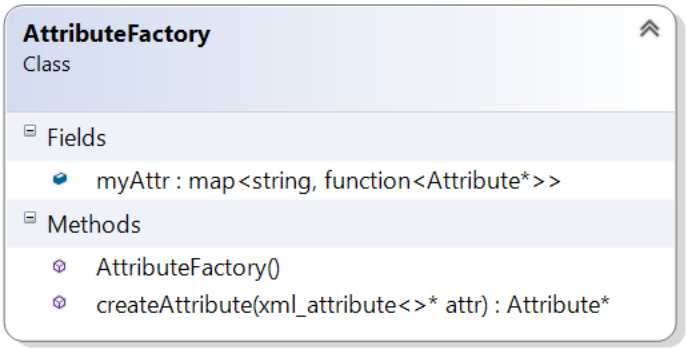
1. Các lớp đối tượng hình học.

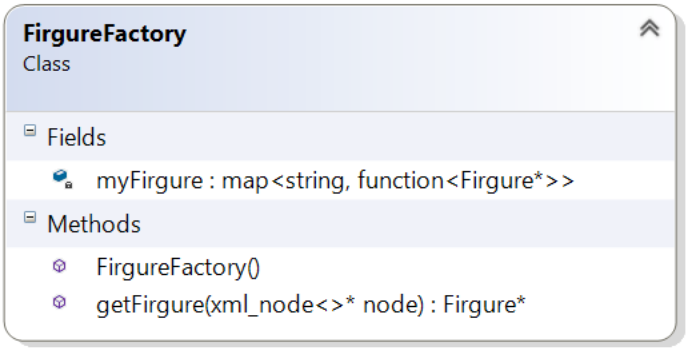


2. Các lớp đối tượng thuộc tính.

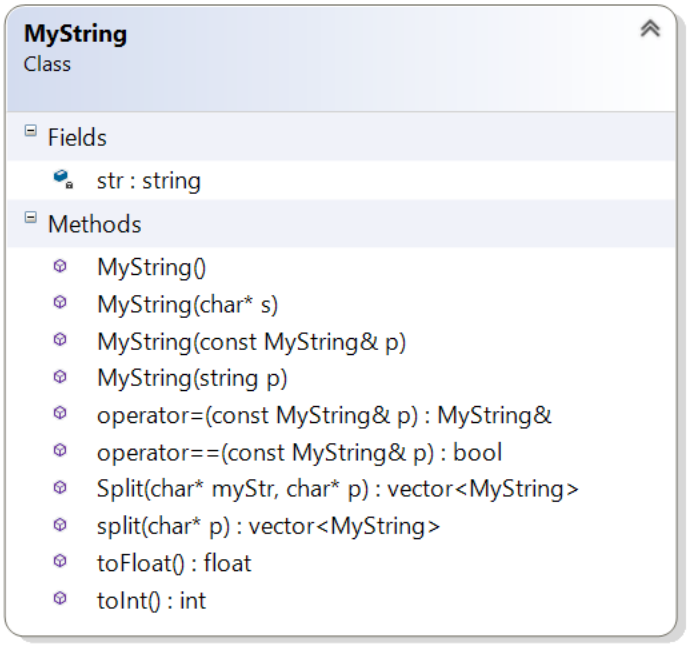


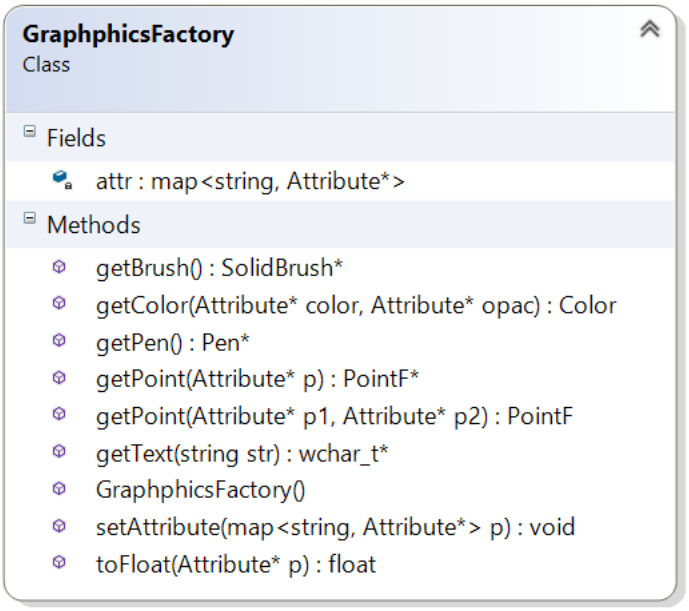
3. Các lớp đối tượng khởi tạo





4. Các lớp đối tượng hỗ trợ





III. Các vấn đề và giải pháp.

1. Vấn đề con trỏ và cấp phát bộ nhớ.

Trong quá trình thiết kế và cài đặt, con trỏ được sử dụng khá nhiều để có thể áp dụng tính chất đa hình trong hướng đối tượng. Vì vậy vấn đề huỷ và giải phóng bộ nhớ được đặt lên hàng đầu. Mỗi đối tượng đều chủ động quản lí bộ nhớ các con trỏ. Các lớp đối tượng có chứa con trỏ đều được cài đặt đầy đủ copy constructor, operator = và destructor.

```
MyPaint(const MyPaint& p)
{
    for (auto i : p.myFirgure)
        myFirgure.push_back(i->clone());
}

~MyPaint()
{
    for (auto i : myFirgure)
        delete i;
    myFirgure.clear();
}

MyPaint& operator = (const MyPaint& p)
{
    if (this != &p)
    {
        for (auto i : myFirgure)
            delete i;
        myFirgure.clear();
        for (auto i : p.myFirgure)
            myFirgure.push_back(i->clone());
    }
    return *this;
}
```

Trong quá trình cài đặt cũng nảy sinh vấn đề copy các đối tượng, vì vậy trong đồ án có áp dụng mẫu Prototype trong việc tạo bản sao của đối tượng.

```
Firgure* CLine::clone() const
{
    return new CLine(*this);
}
```

2. Vấn đề khởi tạo các đối tượng một cách tổng quát.

Để tránh trùng lắp mã nguồn trong quá trình khởi tạo đối tượng, bài làm đã áp dụng mẫu Factory trong việc khởi tạo. Tuy nhiên, mẫu factory trong bài đã có cải tiến, tránh việc sử dụng nhiều câu lệnh if bằng cách vận dụng map của thư viện STL cùng kiểu dữ liệu funcional.

```
class FirgureFactory
{
private:
    map<string, function<Firgure*()>> myFirgure;
public:
    FirgureFactory()
    {
    myFirgure["line"] = []{return new CLine; };
    myFirgure["circle"] = []{return new CCircle; };
    myFirgure["ellipse"] = []{return new CEllipse; };
    myFirgure["rect"] = []{return new CRectangle; };
    myFirgure["polyline"] = []{return new CPolyline; };
    myFirgure["polygon"] = []{return new CPolygon; };
    myFirgure["path"] = []{return new CPath; };
    myFirgure["text"] = []{return new CText; };
    myFirgure["g"] = []{return new CGroup; };
    }

    Firgure* getFirgure(xml_node<> *node)
    {
    return (void*)myFirgure[(string)node->name()]();
    }
};
```

3. Vấn đề xử lí chuỗi đầu vào.

Mặc dù công cụ XML Document của thư viện RapidXML đã xử lí rất tốt chuỗi dữ liệu đầu vào, tuy nhiên trong quá trình cài đặt ta cũng thường xuyên gặp phải vấn đề cắt chuỗi để gán các giá trị cho thuộc tính hình học.

Lớp đối tượng MyString được xây dựng để giải quyết vấn đề này. Lớp MyString có một thuộc tính có kiểu dữ liệu std::string để tận dụng được các hàm có sẵn. Thêm vào đó là lớp MyString có thêm phương thức Split để cắc chuỗi đầu vào theo điều kiện cho trước.
Phương thức Split hoạt động gần giống như phương thức String.Split của C#.

```
vector<MyString> MyString::split(char *p)   const
{
    char *temp = new char[str.length() + 1];
    strcpy_s(temp, str.length() + 1, str.c_str());
    char *s = new char[256];
    vector<MyString> result;
    char *i = NULL;
    s = strtok_s(temp, p, &i);
    if (s != NULL)
        result.push_back((MyString)s);
    while (s = strtok_s(NULL, p, &i))
    {

        result.push_back((MyString)s);
    }
    delete[] temp;
    delete[] s;
    return result;
}
```

Ngoài ra, ta cũng hay gặp phải vấn đề đổi chuỗi sang số. Vì vậy lớp MyString cũng được trang bị thêm phương thức toInt (đổi chuỗi sang số nguyên kiểu int) và toFloat (đổi chuỗi sang số thực kiểu float).

```
float MyString::toFloat() const
{
    return atof(str.c_str());
}

int MyString::toInt() const
{
    return atoi(str.c_str());
}
```

IV. Kết quả.

1. Dữ liệu đầu vào là ảnh sample.svg

2. Dữ liệu đầu vào là ảnh lion.svg

V. Tổng kết.

1. Kinh nghiệm nhận được thông qua quá trình làm đồ án.

- Việc tổ chức các lớp đối tượng.
- Áp dụng thuần thục tính đóng gói, tính kế thừa và tính đa hình.
- Có thêm kinh nghiệm trong việc xử lí con trỏ và cấp phát động.
- Có cơ hội áp dụng 2 mẫu đối tượng phổ biến là Factory và Prototype.
- Có thêm kiến thức về ảnh vector svg, thư viện XMLRapid.

2. Những điều còn chưa làm được.

- Bài làm chỉ làm trên một thư viện đồ hoạ duy nhất là Gdi+.
- Việc cài đặt một số chỗ còn chưa được tối ưu.

Tài liệu tham khảo

- SVG:

  https://www.w3.org/TR/2003/REC-SVG11-20030114/REC-SVG11-20030114.pdf

- GDI+:

  https://msdn.microsoft.com/en-us/library/windows/desktop/ms533798(v=vs.85).aspx

- RapidXML:
  https://github.com/dwd/rapidxml/blob/master/rapidxml.hpp#L1141