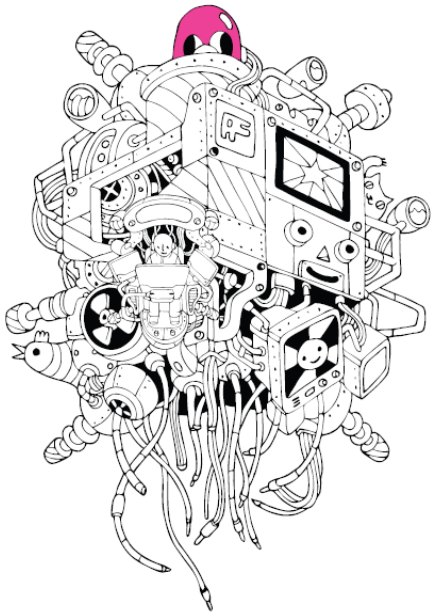
윤성우의
열혈 C++ 프로그래밍



윤성우 저 열혈강의 C++ 프로그래밍 개정판

Chapter 12. String 클래스의 디자인

윤성우의

열혈 C++ 프로그래밍

Chapter 12-1. C++의 표준과 string 클래스

윤성우 저 열혈강의 C++ 프로그래밍 개정판

C++의 새로운 표준안

C++ 0x → C++ 1x

표준 string 클래스

헤더파일 <string>을 포함해야 한다.

```
int main(void)
{
    string str1="I like ";
    string str2="string class";
    string str3=str1+str2;
    cout<<str1<<endl;
    cout<<str2<<endl;
    cout<<str3<<endl;
    str1+=str2;
    if(str1==str3)
        cout<<"동일 문자열!"<<endl;
    else
        cout<<"동일하지 않은 문자열!"<<endl;
    string str4;
    cout<<"문자열 입력: ";
    cin>>str4;
    cout<<"입력한 문자열: "<<str4<<endl;
    return 0;
}
```

string 클래스는 표준 클래스로써 문자열의 처리를 위해서 제공되는 클래스이다!

실행결과

```
I like
string class
I like string class
동일 문자열!
문자열 입력: Hi~
입력한 문자열: Hi~
```

윤성우의

열혈 C++ 프로그래밍

Chapter 12-2. 문자열 처리 클래스의 정의

윤성우 저 열혈강의 C++ 프로그래밍 개정판

표준 string 클래스의 분석

1. 문자열을 인자로 전달받는 생성자의 정의

```
string str1="I like ";          string str1("I like ");
string str2="string class";     string str2("string class");
```

2. 생성자, 소멸자, 복사 생성자, 대입 연산자의 정의

생성자 내에서 문자열 저장을 위한 메모리의 동적 할당이 이루어지므로..

3. 결합된 문자열로 초기화된 객체를 반환하는 + 연산자의 오버로딩

string str3=str1+str2;

str1+str2의 반환 값으로 두 객체의 문자열을 모두 포함하는 string 객체가 반환되도록

4. 문자열을 덧붙이는 += 연산자의 오버로딩

5. 내용비교를 진행하는 == 연산자의 오버로딩

6. 콘솔 입출력을 가능하게 하는 <<, >> 연산자의 오버로딩

String 클래스의 완성1: 클래스의 선언


```
class String
{
private:
    int len;
    char * str;
public:
    String();
    String(const char * s);
    String(const String& s);
    ~String();
    String& operator= (const String& s);
    String& operator+= (const String& s);
    bool operator== (const String& s);
    String operator+ (const String& s);

    friend ostream& operator<< (ostream& os, const String& s);
    friend istream& operator>> (istream& is, String& s);
};
```

String 클래스의 완성2: 복사 생성자, 대입 연산자


```
String::String()
{
    len=0;
    str=NULL;
}
String::String(const char* s)
{
    len=strlen(s)+1;
    str=new char[len];
    strcpy(str, s);
}
String::String(const char* s)
{
    len=strlen(s)+1;
    str=new char[len];
    strcpy(str, s);
}
String::String(const String& s)
{
    len=s.len;
    str=new char[len];
    strcpy(str, s.str);
}
String::~String()
{
    if(str!=NULL)
        delete []str;
}
```

이 생성자는 다음의 형태로 객체생성을 돕는다.

String emptystr;

str이 NULL일수 있으므로 이를 확인하고 delete

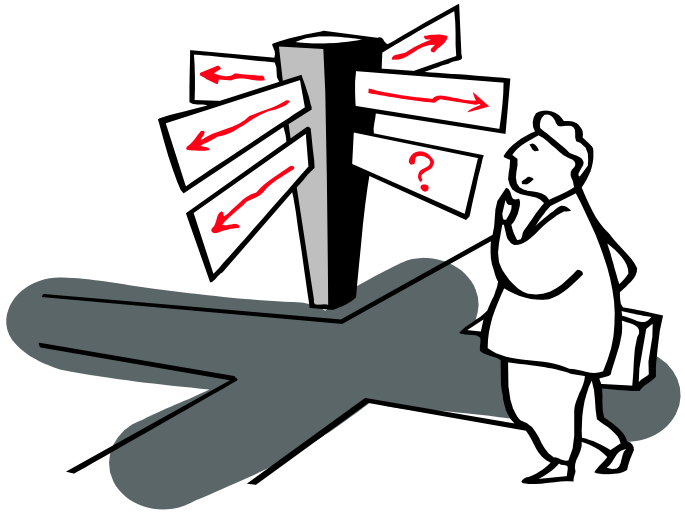
```
String& String::operator= (const String& s)
{
    if(str!=NULL)
        delete []str;
    len=s.len;
    str=new char[len];
    strcpy(str, s.str);
    return *this;
}
String& String::operator+= (const String& s)
{
    len+=(s.len-1);
    char* tempstr=new char[len];
    strcpy(tempstr, str);
    strcat(tempstr, s.str);

    if(str!=NULL)
        delete []str;
    str=tempstr;
    return *this;
}
```

배열은 확장이 불가능하므로, 새로운 배열을 생성하고 기존 배열 삭제

String 클래스의 완성3: 나머지 멤버함수들


```
String String::operator+ (const String& s)
{
    char* tempstr=new char[len+s.len-1];
    strcpy(tempstr, str);
    strcat(tempstr, s.str);
    String temp(tempstr);  // String temp=tempstr;
    delete []tempstr;
    return temp;
}
ostream& operator<< (ostream& os, const String& s)
{
    os<<s.str;
    return os;
}
istream& operator>> (istream& is, String& s)
{
    char str[100];
    is>>str;
    s=String(str);
    return is;
}
```

```
bool String::operator== (const String& s)
{
    return strcmp(str, s.str) ? false : true;
}
```



Chapter 12가 끝났습니다. 질문 있으신지요?