C++ 프로그래밍 실습

Visual Studio 2015

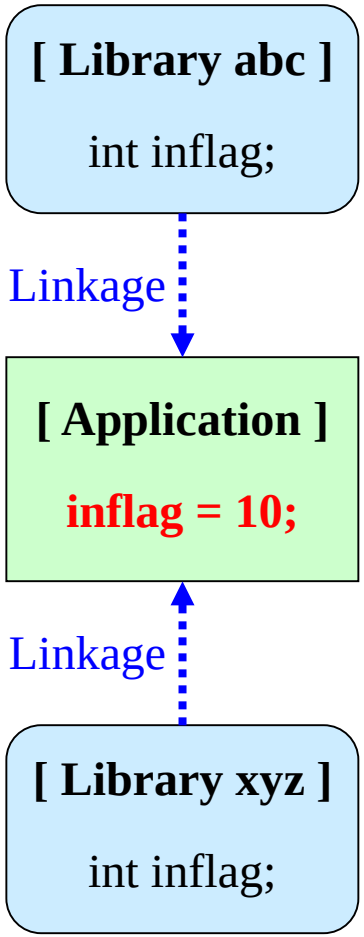
Contents

- Namespaces
- I/O Stream
- Exercise

- **Practice1 -Namespaces**

Practice 1 : Namespaces

- Name conflict problem
  - Occurs when an application tries to use more than two libraries that have an identifier with the same name
  - A potential problem in large applications involving several programmers (must be careful to find unique names)



- C++ provides ***namespaces*** to prevent name conflicts
  - By disambiguating a name using the scope resolution operator "**::**"
  - E.g., identifiers in the standard C++ libraries are covered by the namespace "**std**" (i.e., **cout** in the **iostream** library is actually identified as **std::cout**)

Practice 1 : Namespaces

```
#include <iostream>

int main(void) {
    ...
    cout << "Test";
    ...
}
```

- Valid C++ programs

```
#include <iostream>

int main(void) {
    ...
    std::cout << "Test";
    ...
}
```

(a) Scope resolution OP

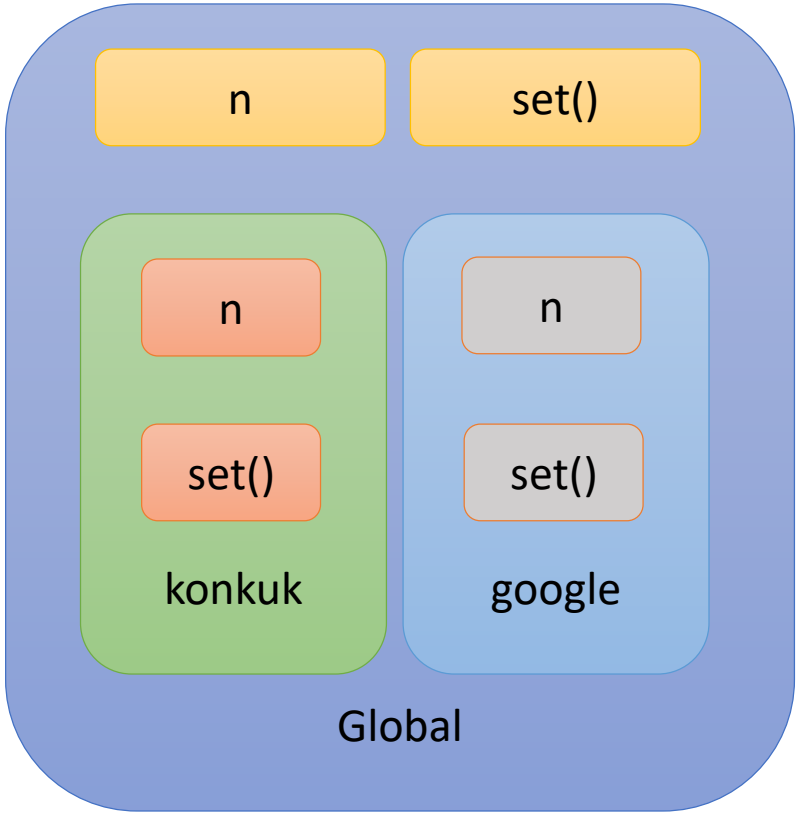
```
#include <iostream>
using std::cout;

int main(void) {
    ...
    cout << "Test";
    ...
}
```

(b) Using declaration

```
#include <iostream>
using namespace std;

int main(void) {
    ...
    cout << "Test";
    ...
}
```

(c) Using directive

Practice 1-1 : Namespaces



Practice 1-1 : Namespaces

```
#include<iostream>

using namespace std; //std 이름공간에 선언된 모든 이름에 std:: 생략

int n;
void set() {
    ::n = 10; //명시적 전역변수(아무런 namespace에 속해 있지 않다.)
}

namespace konkuk {
    int n;
    void set() {
        konkuk::n = 20; //namespace 명시하지 않아도 됨.
    }
}
namespace google {
    int n;
    void set() {
        google::n = 30; //namespace 명시하지 않아도 됨.

    }
}
```

Practice 1-1 : Namespaces

```
int main() {
    ::set();
    konkuk::set();
    google::set();

    cout << ::n << endl;
    cout << konkuk::n << endl;
    cout << google::n << endl;
}
```

Execution Result:

```
10
20
30
```

Practice 1-2 : Openness of Namespaces

```
namespace A {
    int f();
}

namespace A {
    int g();
    int h();
}
```

=

```
namespace A {
    int f();
    int g();
    int h();
}
```

(a) Separate namespaces

(b) Single integrated namespace

Practice 1-2 : Openness of Namespaces

```
#include<iostream>

using namespace std;

int n;
void set();

namespace konkuk {
    int n;
    void set();
}

namespace google {
    int n;
    void set();
}
```

Practice 1-2 : Openness of Namespaces

```
int main() {
   ::set();
   konkuk::set();
   google::set();

   cout << ::n << endl;
   cout << konkuk::n << endl;
   cout << google::n << endl;
}
```

Execution Result:

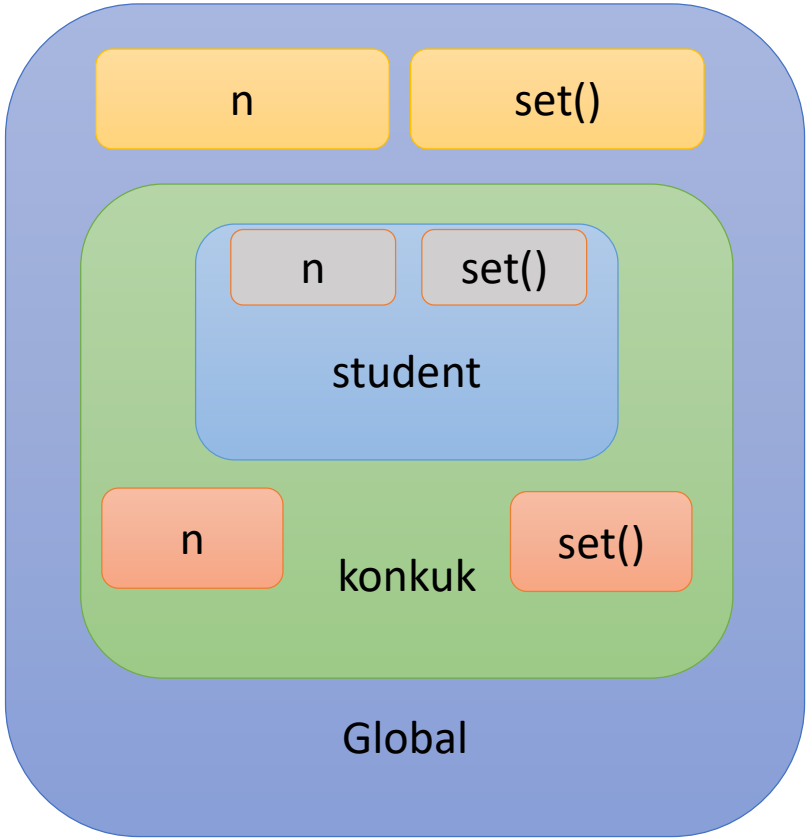
```
10
20
30
```

```
void set() {
   n = 10;
}
namespace konkuk {
   void set() {
      n = 20;
   }
}

namespace google {
   void set() {
      n = 30;
   }
}
```

=

```
void ::set(){
   n = 10;}


void konkuk::set(){
   n = 20;
}

void google::set(){
   n = 30;
}
```

Practice 1-3 : Nested Namespaces



Practice 1-3 : Nested Namespaces

```
#include<iostream>

using namespace std;

int n;

void set() {
    n = 10;
}

namespace konkuk {
    int n;
    void set() {
        n = 20;
    }
    namespace student {
        int n;
        void set() {
        n = 30;
        }
    }
}
```

Practice 1-3 : Nested Namespaces

```
int main() {
    ::set();
    konkuk::set();
    konkuk::student::set();

    cout << ::n << endl;
    cout << konkuk::n << endl;
    cout << konkuk::google::n << endl;

    return 0;
}
```

Execution Result:

```
10
20
30
```

- **Practice2 –I/O stream**

Practice 2 : I/O Stream

- C++ provides an alternative to I/O library of C
  - Easier-to-use, extensible, and more flexible
  - This section introduces basic I/O functionalities of C++
    - The details on C++ I/O are covered in Chap. 8

- ***Stream-based I/O*** of C++
  - Input to a C++ program is treated as a ***stream*** of consecutive bytes from an input device (e.g., keyboard, disk, scanner, …)
  - Output from a C++ program is also treated as a ***stream*** of consecutive bytes to an output device (e.g., video display, disk, printer, …)

Practice 2 : I/O Stream

- The standard I/O variables are used with the I/O operators
  - Input operator "**>>**"
    - E.g., **`cin >> x;`** // reads a value from the keyboard and store the value into ***x***
  - Output operator "**<<**"
    - E.g., **`cout << x;`** // writes the value of ***x*** to the display

- I/O operators
  - Left-associated
    - Evaluated from left to right
    - E.g., **`cout << x << y;`** // writes ***x*** first and then ***y*** to the display
  - Automatically recognize type of the data
    - No format string is required (cf., **printf** or **scanf** requires a format string)
    - E.g.,

```
int x;
cin >> x;          //  == scanf("%d", &x)
cout << x;         //  == printf("%d", x)
```

Practice 2-1 : I/O Stream

```
#include<iostream>
using namespace std;

int main() {
   int width;
   cout << "너비를 입력하세요: ";
   cin >> width;

   int height;
   cout << "높이를 입력하세요: ";
   cin >> height;

   int area = width*height;

   cout << "면적: " << area << endl;

}
```

Execution Result:

```
너비를 입력하세요:3
높이를 입력하세요:5
면적: 15
```

Practice 2-2 : I/O Stream

- It is possible to read from and write to (disk) files in the same way as using **`cin`** and **`cout`**
  - At first, "**`fstream`**" header needs to be included
  - Then, we can replace
    - **`cin`** with an object of **`ifstream`** class associated with the input file, and
    - **`cout`** with an object of **`ofstream`** class associated with the output file
  - The I/O operators "**>>**" and "**<<**" are used in just the same way as they are used with **`cin`** and **`cout`**

Practice 2-2 : I/O Stream

```
#include <iostream>
#include<string>
#include <fstream> //입출력 헤더
using namespace std;

int main() {
   ofstream ofs; //ofstream 클래스의 객체 선언
   string str = "친구3명의 이름과 나이를 입력하세요.";

   ofs.open("friend.txt");  //파일열기

   if (!ofs) {
      cout << "파일을 열 수 없다." << endl;
      exit(0); }

   cout << str << endl;
   for (int i = 0; i<3; i++) {
      int Age;
      string Name;
      cout << "이름: ";
      cin >> Name;
      cout << "나이 : ";
      cin >> Age;
      ofs << Name << " " << Age << endl;  //출력파일 스트림에 기록
   }
   ofs.close(); //파일 닫기

   //fileLoad();
}
```

Reads a values from the keyboard and store the value into ***Name and Age***

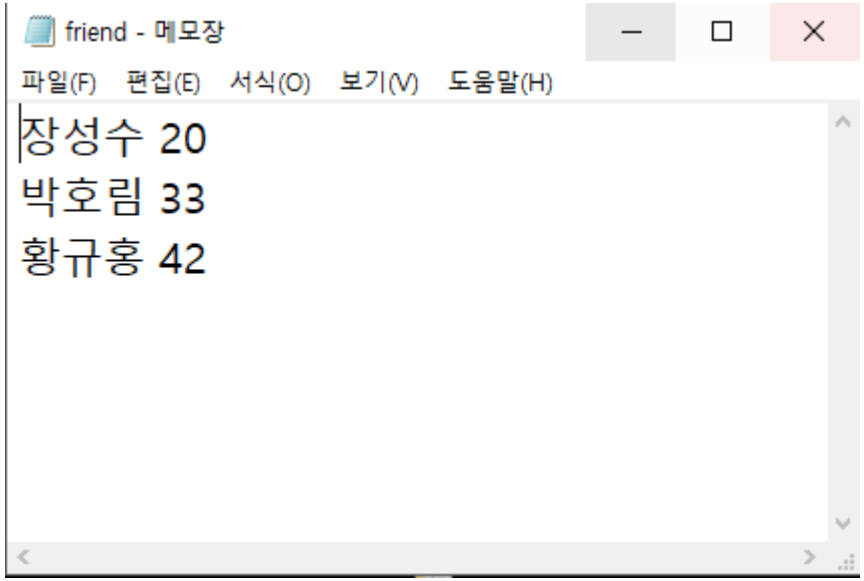
Practice 2-2 : I/O Stream

```
void fileLoad() {
    int Age;
    string Name;
    string fileName = "friend.txt";
    ifstream ifs; //ifstream 클래스의 객체 선언

    ifs.open(fileName.c_str()); //파일열기

    if (!ifs) {
    cout << "파일을 열 수 없다." << endl;
    exit(0); }

    cout << "\n==List of your friends from the stored file==\n";

    while (ifs >> Name >> Age) {
        cout << Name << " " << Age << endl;
    }
    ifs.close(); //파일닫기
}
```

입력 파일 스트림에서 읽기

Practice 2-2 : I/O Stream

```
친구3명의 이름과 나이를 입력하세요.
이름: 장성수
나이 : 20
이름: 박호림
나이 : 33
이름: 황규홍
나이 : 42

==List of your friends from the stored file==
장성수 20
박호림 33
황규홍 42
계속하려면 아무 키나 누르십시오 . . .
```



Exercise-1

- 자유롭게 Namespaces를 설계하시오
  - 최소 3개의 Namespaces를 포함한다.
  - Nested Namespaces를 사용한다.

Exercise-2

- 자신의 이름, 나이 ,메일을 입력하여 텍스트 파일로 출력하시오.
  -출력된 텍스트 파일을 읽어 콘솔창에 출력

Course Homepage

- How to access
  - URL: sclab.konkuk.ac.kr
- Downloading class material
  - Students can download syllabus and lecture notes in PDF format
- Class announcement
  - About homework and project
  - Exam schedule and result
  - And so on

Submit

- Teaching assistant: 장성수
  Office: 신공학관 1216호 (대학원 SCLab 연구실)
  Email: pik1100@naver.com

- Title of the email : [2018][Practice#]_student# _ student _ name

- Ex) [2018][Practice02]_201700000_장성수

- Create zip file. (C++ project folder)

- 주의 : 메일 양식이 잘못될 경우 채점이 되지 않을 수 있음.

- 질문 메일 : pik1100@naver.com : 장성수

끝