윈도우프로그래밍

2. C++ 기초(2)

2018. 3.16.

심미나 교수

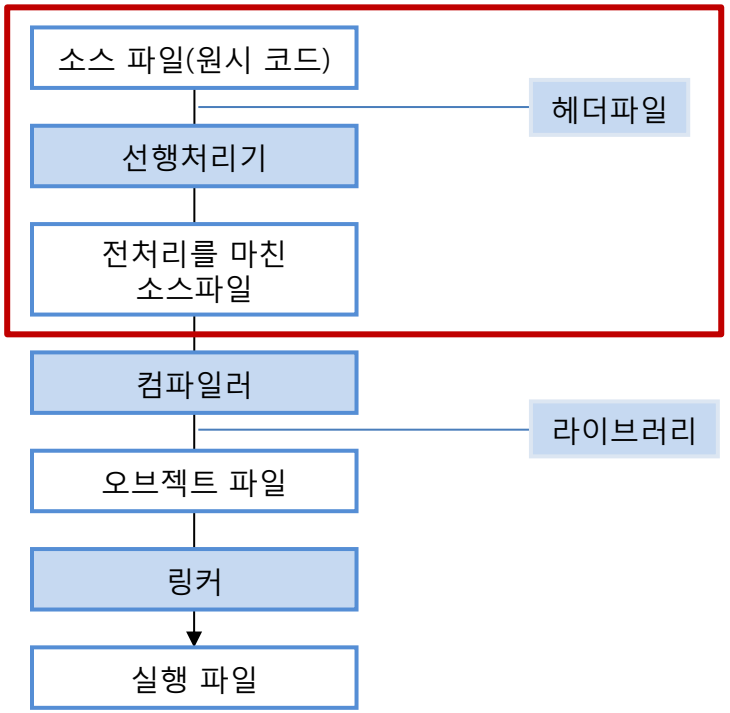
목 차

I. C와 C++ 비교(2)

C와 C++ 비교

C++의 인라인(inline) 함수

- (참고) 선행처리기
  - 컴파일러에서 컴파일을 하기에 앞서 소스 파일에 대한 처리를 선행하는 것
  - #으로 시작하는 선행처리의 명령어를 찾아 해당 작업을 통해 원시코드 내용 변경
    - (ex) #define, #undef, #include, #in, #indef, #elif, #else, #endif, #line 등
    - #include의 경우, 이후에 나오는 파일을 찾아 그 파일에 기술된 내용을 현재 파일의 #include 기술부분에 대체



C와 C++ 비교

C++의 인라인(inline) 함수

- **(참고) 매크로 함수**

(형식) #define 매크로함수(매개변수목록) 치환할 내용(문자열/상수)

  - #define문을 이용하여 위와 같이 정의되며, 선행처리기에 의해 처리되는 함수
  - 매크로함수가 호출되면, 그 위치에 매크로 정의 시 지정된 문자열/상수를 치환
  - (특징)
    - 일반함수는 호출 함수를 찾아가서 수행 후 돌아오지만 매크로함수는 소스부분을 확장하는 개념이므로 프로그램의 처리 속도를 개선할 수 있음
    - 단, 함수 정의가 복잡한 경우에는 한계 존재

```
#define SQUARE(x) ((x)*(x))

int main(void)
{
    std::cout<< SQUARE(5) <<std::endl;
    return 0;
}
```

선행처리 결과

```
int main(void)
{
    std::cout<< ((5)*(5)) <<std::endl;
    return 0;
}
```

C와 C++ 비교

C++의 인라인(inline) 함수

- **(참고) 매크로 함수**

(형식) #define 매크로함수(매개변수목록) 치환할 내용(문자열/상수)

  - **(특징)**
    - **매크로함수는 자료형에 독립적**

      (예시)

```
#define SQUARE(x) ((x)*(x))
```

    - std:cout<< SQUARE(12); **과 같은** int**형 함수호출이나,**
    - std::cout<< SQUARE(3.15); **과 같은** double**형 함수호출이 가능 즉, 자료형에 독립적임**

C와 C++ 비교

C++의 인라인(inline) 함수

- 매크로와 기능이 유사한 특수함수로 컴파일러에 의해 처리
  - 매크로 함수의 매크로 함수의 인라인화로 인한 장점(성능향상)은 그대로 가지면서 매크로 함수의 단점(복잡한 함수정의 한계)을 극복하기 위한 것
- 다음과 같이 일반함수 정의와 동일하며 앞에 키워드 inline만 추가

```
(형식) inline  반환자료형  함수명  (매개변수목록)
       {
           변수 선언;
           문장;
           return (반환값);
        }
```

- 인라인 함수 호출 시, 호출부분에 인라인 함수를 대체
  - 즉, 함수가 아닌 일반구문 처리되며, 함수 호출 시 스택의 변화가 없음
  - 키워드 inline선언은 컴파일러에 의해 처리됨(컴파일러가 함수의 인라인화 결정)

C와 C++ 비교

C++의 인라인(inline) 함수

- (EX2-01) 인라인함수 사용

EX2-01

```
inline int SQUARE(int x)
{
    return x*x;
}

int main(void)
{
    std::cout<<SQUARE(5)<<std::endl;
    std::cout<<SQUARE(12)<<std::endl;
    return 0;
}
```

실행 결과

```
25
144
```

(예시)

```
std:cout<< SQUARE(12);
std:cout<< SQUARE(3.15);
```

- Inline 함수를 위의 형태로 호출하려면 각 자료형별로 함수가 오버로딩되어야 함
- 즉, 매크로함수와 달리 자료형에 독립적이지 않음(단점)
  - 이를 개선한 것이 '템플릿'

```
template <typename T>
inline T SQUARE(T x)
{
    return x*x;
}
```

C와 C++ 비교

C++의 이름공간(namespace)

- ‘이름충돌’ 문제를 방지하기 위한 것
  - C++이 지원하는 각종 요소들(변수, 함수, 클래스 등)을 한 범주로 묶어주는 기능
  - 회사나 팀에서 코드를 함께 사용하는 프로그램을 쉽게 작성하게 해줌
    - 여러 회사에서 함수명 등을 사용할 때 동일한 이름을 사용하게 되어 ‘충돌’이 발생하는 것을 방지하기 위한 것
  - (형식) 정의: 키워드 namespace ‘이름공간명칭’
    사용: ‘사용할 객체가 속한 이름공간’ ::(범위지정연산자) ‘객체’

※ 공간을 ‘소속’ 또는 ‘범주’로 이해하면 편리

C와 C++ 비교

C++의 이름공간(namespace)

- **존재하는 이름공간이 다르면 동일한 이름의 함수 및 변수 선언 가능**

EX2-02

```
namespace BestComImpl
{
    void SimpleFunc(void)
    {
        std::cout<<"BestCom이 정의한 함수"<<std::endl;
    }
}
namespace ProgComImpl
{
    void SimpleFunc(void)
    {
        std::cout<<"ProgCom이 정의한 함수"<<std::endl;
    }
}
int main(void)
{
    BestComImpl::SimpleFunc();
    ProgComImpl::SimpleFunc();
    return 0;
}
```

실행 결과

```
BestCom이 정의한 함수
ProgCom이 정의한 함수
```

ⓘ 이름공간 BestComImpl에 정의된 SimapleFunc 호출

→ 'BestComImpel'에 존재하는 SimpleFucn()함수 호출

ⓘ 이름공간 ProgComImpl에 정의된 SimapleFunc 호출

→ 'ProgComImpl'에 존재하는 SimapleFunc ()함수호출

C와 C++ 비교

C++의 이름공간(namespace)

- 이름공간에 속한 함수의 선언과 정의
  - 함수선언: 반드시 이름공간 안에 선언
  - 함수정의: 반드시 범위지정함수명 사용

EX2-03

ⓘ 이름공간에 속한 함수의 선언

```
namespace BestComImpl
{
    void SimpleFunc(void);
}
namespace ProgComImpl
{
    void SimpleFunc(void);
}
int main(void)
{
    BestComImpl::SimpleFunc();
    ProgComImpl::SimpleFunc();
    return 0;
}
```

ⓘ 이름공간에 속한 함수의 정의

```
void BestComImpl::SimpleFunc(void)
{
    std::cout<<"BestCom이 정의한 함수"<<std::endl;
}
void ProgComImpl::SimpleFunc(void)
{
    std::cout<<"ProgCom이 정의한 함수"<<std::endl;
}
```

ⓘ 이름공간 BestComImpl에 정의된 SimapleFunc의 선언과 정의의 분리

ⓘ 이름공간 ProgComImpl에 정의된 SimapleFunc의 선언과 정의의 분리

실행 결과

C와 C++ 비교

C++의 이름공간(namespace)

- **동일한 이름공간 내에서의 함수호출**
  - 선언된 '이름공간의 이름'이 동일하면, 이렇게 선언된 함수들은 '이름공간'이 동일한 것으로 간주
  - 이름공간을 지정하지 않으면, 호출문이 존재하는 함수와 동일한 이름공간 사용

ⓘ 동일하게 BestComImple 소속 함수

```
namespace BestComImpl
{
    void SimpleFunc(void);
}
namespace BestComImpl
{
    void PrettyFunc(void);
}
```

ⓘ 동일하게 BestComImple 소속 함수

```
void BestComImpl::SimpleFunc(void)
{
    std::cout<<"BestCom이 정의한 함수"<<std::endl;
    PrettyFunc();                    // 동일 이름공간
    ProgComImpl::SimpleFunc();       // 다른 이름공간
}

void BestComImpl::PrettyFunc(void)
{
    std::cout<<"So Pretty!!"<<std::endl;
}
```

C와 C++ 비교

C++의 이름공간(namespace)

실행 결과

- (EX2-04) 동일한 이름공간 내에서의 함수호출

```
#include <iostream>

namespace BestComImpl
{
    void SimpleFunc(void);
}

namespace BestComImpl
{
    void PrettyFunc(void);
}

namespace ProgComImpl
{
    void SimpleFunc(void);
}
```

```
int main(void)
{
    BestComImpl::SimpleFunc();
    return 0;
}

void BestComImpl::SimpleFunc(void)
{
    std::cout<<"BestCom이 정의한 함수"<<std::endl;
    PrettyFunc();
    ProgComImpl::SimpleFunc();
}

void BestComImpl::PrettyFunc(void)
{
    std::cout<<"So Pretty!!"<<std::endl;
}

void ProgComImpl::SimpleFunc(void)
{
    std::cout<<"ProgCom이 정의한 함수"<<std::endl;
}
```

윈도우프

C와 C++ 비교

C++의 이름공간(namespace)

- **이름공간의 중첩**
  - 이름공간은 중첩이 가능하므로 계층적 구조를 가질 수 있고, 체계적 구분이 가능

```
namespace Parent
{
    int num=2;
    namespace SubOne
    {
        int num=3;
    }
    namespace SubTwo
    {
        int num=4;
    }
}
```

ⓘ 이름공간 Parent

ⓘParent::num

ⓘ 이름공간 Parent::SubOne

ⓘParent::SubOne::num

ⓘ 이름공간 Parent::SubTwo

ⓘParent::SubTwo::num

C와 C++ 비교

C++의 이름공간(namespace)

- C++ 표준요소들의 이름공간, std
  - C++표준에서 제공하는 다양한 요소들은 이름공간 std 내에 선언되어 있음

(예) <iostream>에 선언된 cout, cin, endl

- std::cout → '이름공간 std에 선언된 cout'
- std::cin → '이름공간 std에 선언된 cin'
- std::endl → '이름공간 std에 선언된 endl'

```
namespace std
{
    cout . . . .
    cin . . . .
    endl . . . .
}
```

C와 C++ 비교

C++의 이름공간(namespace)

- using 예약어를 이용한 이름공간 선언
  - 프로그램 내부에서 자주 사용하는 이름공간의 경우, 식별자 앞에 기술하지 않고 생략함으로써 간단하게 사용할 수 있도록 하는 것
  - 단, using namesapce의 빈번한 사용은 이름충돌을 막기 위한 '이름공간 선언'의 의미를 퇴색시키므로 제한적으로 사용해야 함

EX2-05

```
#include <iostream>
namespace Hybrid
{
    void HybFunc(void)
    {
        std::cout<<"So simple function!"<<std::endl;
        std::cout<<"In namespace Hybrid!"<<std::endl;
    }
}
int main(void)
{
    using Hybrid::HybFunc;
    HybFunc();
    return 0;
}
```

실행 결과

```
So simple function!
In namespace Hybrid!
```

C와 C++ 비교

C++의 이름공간(namespace)

- using 을 이용한 이름공간 선언

EX2-06

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

int main(void)
{
    int num=20;
    cout<<"Hello World!"<<endl;
    cout<<"Hello "<<"World!"<<endl;
    cout<<num<<' '<<'A';
    cout<<' '<<3.14<<endl;
    return 0;
}
```

이후 Std:cin 대신 cin 사용

EX2-07

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

int main(void)
{
    int num=20;
    cout<<"Hello World!"<<endl;
    cout<<"Hello "<<"World!"<<endl;
    cout<<num<<' '<<'A';
    cout<<' '<<3.14<<endl;
    return 0;
}
```

이름공간 std 에 선언된 모든 것에 대해 이름공간 std 생략

실행 결과

C와 C++ 비교

C++의 이름공간(namespace)

- **이름공간의 별칭**
  - 다음과 같은 형식으로 이름공간의 별칭 선언

    Namespace ABC=AAA::BBB:::CCC;
    - AAA::BBB::CCC 에 대해 ABC라는 별칭 선언

  - 다음 예시처럼 별칭으로 이름공간의 선언을 대신함
    - ABC::num1 = 1;
    - ABC::num2 = 20;

```
namespace AAA
{
    namespace BBB
    {
        namespace CCC
        {
            int num1;
            int num2;
        }
    }
}
```

C와 C++ 비교

C++의 이름공간(namespace)

- (EX2-08) 이름공간의 별칭

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

namespace AAA
{
    namespace BBB
    {
        namespace CCC
        {
            int num1;
            int num2;
        }
    }
}
```

```
int main(void)
{
    AAA::BBB::CCC::num1=20;
    AAA::BBB::CCC::num2=30;

    namespace ABC=AAA::BBB::CCC;

    cout<<ABC::num1<<endl;
    cout<<ABC::num2<<endl;
    return 0;
}
```

실행 결과

C와 C++ 비교

C++의 이름공간(namespace)

- **범위지정연산자의 부가기능 - 전역변수의 접근**
  - 전역변수에 접근할 때 범위지정연산자(::)를 사용하면 편리
  - 즉, (이름공간이 없는) 전역변수 접근을 의미

```
int val=100;          // 전역변수

int SimpleFunc(void)
{
    int val=20;       // 지역변수
    val+=3;           // 지역변수 val의 값 3 증가
    ::val+=7;         // 전역변수 val의 값 7 증가
}
```

II. C와 C++ 비교(3)

C와 C++ 비교

C++의 자료형 Bool

- '참'을 의미하는 true, '거짓'을 의미하는 false
  - #define TRUE 1, #define FALSE 0 정의 → while (TRUE) { 반복할 내용 }
  - 대신, '참'과 '거짓'을 의미하는 문자키워드 true와 false 사용
  - 각각 정수 0과 1을 의미하는 1바이트 데이터 (4바이트 정수가 아님!)
  - (예시)

    Int num1 = true; (num1에 1 저장)

    Int num2 = false; (num2에 0 저장)

    Int num3 = true+false (num3=1+0)

EX2-09

```
int main(void)
{
    int num=10;
    int i=0;
    cout<<"true: "<<true<<endl;
    cout<<"false: "<<false<<endl;
    while(true)
    {
        cout<<i++<<' ';
        if(i>num)
            break;
    }
    cout<<endl;
    cout<<"sizeof 1: "<<sizeof(1)<<endl;
    cout<<"sizeof 0: "<<sizeof(0)<<endl;
    cout<<"sizeof true: "<<sizeof(true)<<endl;
    cout<<"sizeof false: "<<sizeof(false)<<endl;
    return 0;
}
```

ⓘ 정수

ⓘ 1과0으로 대체된 값
ⓘ (1byte)

실행 결과

```
true:
false:
sizeof 1:
sizeof 0:
sizeof true:
sizeof false:
```

C와 C++ 비교

C++의 자료형 Bool

- **기본개념**
  - true와 false는 bool형 데이터
  - true와 false 정보를 저장할 수 있는 변수, bool형 변수
    - Bool형 변수는 '1', '0'을 저장하기 위한 것이 아닌, 'true', 'false'를 저장하기 위한 공간
    - (예시)

```
bool isTrueOne=true;
bool isTrueTwo=flase;
```

ⓘ 반환형

```
bool IsPositive(int num)
{
    if(num<0)
        return false;
    else
        return true;
}
```

EX2-10

```
int main(void)
{
    bool isPos;
    int num;
    cout<<"Input number: ";
    cin>>num;
    isPos=IsPositive(num);
    if(isPos)
        cout<<"Positive number"<<endl;
    else
        cout<<"Negative number"<<endl;

    return 0;
}
```

실행 결과

```
Input number: 12
Positive number
```

C와 C++ 비교

const

- [문제1] 키워드 const는 어떠한 의미를 갖는가?
  다음 문장들을 대상으로 이를 설명하시오.

```
const int num = 10;

const int * ptr1 = &val1;

int * const ptr2 = &val2;

const int * const ptr3 = &val3;
```

C와 C++ 비교

메모리공간

- [문제2] 실행중인 프로그램의 메모리 공간
  - 실행중인 프로그램은 운영체제로부터 메모리 공간을 할당받는다.
  - 이는 크게 데이터, 스택, 힙 영역으로 나뉘는데, 각각의 영역에는 어떠한 형태의 변수가 할당되는지 설명하시오.

C와 C++ 비교

Call-by-value, Call-by-reference

- [문제3] Call-by-value, Call-by-reference
  - 함수의 호출형태는 크게 '값에 의한 호출'과 '참조에 의한 호출'로 나뉜다.
  - 이 둘을 나누는 기준이 무엇인지, swap함수(두 int형 변수의 값을 교환하는 함수)를 예로 들어 설명하시오.

```
Void SwapByValue(                    )
{


}

//Call-by-value
```

```
Void SwapByRef(                      )
{


}

//Call-by-reference
```

IV. 실습

실습2-01

교재 - 예제 9-2,9-3.(p.334-336)

제곱 승을 구하는 매크로 함수의 오류 수정하기

```
01 #include <iostream>
02 using namespace std;
03 #define SQUARE1(x) x*x
04 void main(void)
05 {
06 int a=5, res;
07 res = SQUARE1(a+2);
08 cout << " SQUARE1(a+2) => " << res << "₩n"; // 17
09
10 res = 100/SQUARE1(a);
11 cout << " 100/SQUARE1(a) => " << res << "₩n"; // 100
12 }
```

※ 7, 10행의 박스 내용을 적용할 경우 결과값의 오류가 발생할 수 있음. 이를 고려하여 3행을 수정할 것

실습2-02

교재 - 예제 9-4.(p.337)

매크로 함수, 인라인 함수, 일반 함수의 차이점 살펴보기

```
01 #include <iostream>
02 using namespace std;
03 #define add1(a, b) a+b
04 inline int add2(int a, int b)
05 {
06 return a+b;
07 }
08 int add3(int a, int b)
09 {
10 return a+b;
11 }
12 void main(
13 {
14 int result;
15 result = 2*add1(10, 20);
16 cout<<"macro 함수 => "<<result<<"₩n";
17 result = 2*add2(10, 20);
18 cout<<"inline 함수 => "<<result<<"₩n";
19 result = 2*add3(10, 20);
20 cout<<"일반 함수 => "<<result<<"₩n";
21 }
```

using 구문으로 네임스페이스 사용하기

```
01 #include <iostream> // 헤더파일을 포함시키는 문장
02 using namespace std; // 네임스페이스를 지정
03 void main()
04 {
05 cout<<" 이 름 : OOO " <<endl;
06 cout<<" 소 속 : OOOOOO " <<endl;
07 cout<<" 이메일 : OO@OOOOOO "<<endl;
08 }
```

값에 의한 전달 방식의 함수 익히기

```
01 #include <iostream>
02 using namespace std;
03 int add(int x, int y);
04 void main()
05 {
06 int a=10, b=20, sum;
07 sum=add(a, b);
08 cout<<" sum = "<< sum <<"₩n";
09 }
10 int add(int x, int y)
11 {
12 int z;
13 z=x+y;
14 return(z);
15 }
```

실습2-05

교재 - 예제 5-6.(p.200)

두 변수에 저장된 값 교환하기

```
01 #include <iostream>
02 using namespace std;
03 void main()
04 {
05 int a=10, b=20;
06 cout<<" a => "<< a <<" b => "<< b <<"₩n";
07 int t;
08 t=a;
09 a=b;
10 b=t;
11 cout<<" a => "<< a <<" b => "<< b <<"₩n";
12 }
```

실습2-06

교재 - 예제 5-7.(p.201)

값에 의한 전달 방식으로 두 변수값을 교환하는 함수

```
01 #include <iostream>
02 using namespace std;
03 void swap(int a, int b);
04 void main()
05 {
06 int a=10, b=20;
07 cout<<" a => "<< a <<" b => "<< b <<"₩n";
08 swap(a, b);
09 cout<<" a => "<< a <<" b => "<< b <<"₩n";
10 }
11 void swap(int a, int b)
12 {
13 int t;
14 t=a;
15 a=b;
16 b=t;
17 }
```

실습2-07

교재 - 예제 5-8.(p.202)

주소에 의한 전달 방식으로 두 변수값을 교환하는 함수 작성하기

```
01 #include <iostream>
02 using namespace std;
03 void swap(int *pa, int *pb);
04 void main()
05 {
06 int a=10, b=20;
07 cout<<" a => "<< a <<" b => "<< b <<"₩n";
08 swap(&a, &b);
09 cout<<" a => "<< a <<" b => "<< b <<"₩n";
10 }
11 void swap(int *pa, int *pb)
12 {
13 int t;
14 t=*pa;
15 *pa=*pb;
16 *pb=t;
17 }
```

과제2

파일명 “과제2_분반_학번_이름”으로 제출

- **과제2**
  - 2-1. 교안의 [문제1] 정리하기
  - 2-2. 교안의 [문제2] 정리하기
  - 2-3. 교안의 [문제3] 정리하기

- **제출 시 주의사항**
  - 보고서(HWP, MS Word) 파일 작성
  - 겉표지에 “학과, 학년, 분반, 학번, 이름” 반드시 기재할 것
  - 분량: 겉표지 포함 4페이지 (A4 11포인트)

감사합니다

mnshim@sungkyul.ac.kr