1. C++ 프로그램 시작

C++은 C의 명령형 특징을 일부 향상시켰고, 객체지향 프로그래밍 개념을 도입하였고, 시간이 지나면서 참조형, 인라인 함수, 디폴트 매개변수, 가상함수, 예외처리, 템플릿 등이 추가되었다.

C++은 독립적인 함수를 지원하므로 절차적 프로그래밍이 가능하고, 메서드를 지원하므로 객체지향 프로그래밍도 가능하다. C++에서는 메서드를 보통 멤버함수라고 지칭한다.

C++은 C와 거의 호환이 가능하다. 대부분의 C 프로그램은 C++ 프로그램으로 번역될 수 있다.

// C++ 프로그램 예

```
#include <iostream>     //최신 방식
using namespace std;    //최신 방식
void main()
{
  int a = 5, y;
  y = - a++;
  cout<<a<<", "<<y<<endl;
}
```

```
#include <iostream.h>     //구식 방식
void main()
{
  int a = 5, y;
  y = - a++;
  cout<<a<<", "<<y<<endl;
}
```

[실행결과]

6, -5

// 해설 ----------

y = -a++;

↳ 먼저, -a가 y에 대입되고, 나중에 a가 1 증가된다.

만약, **y = -++a;**이면 6, -6이 출력된다.

// 이름 공간

프로그래밍 언어에서 함수, 변수, 상수 등은 이름을 통해서 참조된다.
C++에서 이름 공간은 이들 이름의 사용 범위를 보다 효과적으로 제어하기 위해 제공된다.

```
using namespace std;
```

- std라는 이름 공간에 정의되어 있는 모든 이름들을 프로그램 내에서 사용할 수 있도록 하겠다는 뜻이다. 즉, 접두어 std::를 붙이지 않고 std에 정의된 이름을 사용할 수 있다.
- 이름 공간은 키워드 namespace를 이용하여 정의한다.
- using은 이름 공간에 접근할 수 있도록 한다.

```
#include <iostream>
using namespace std;
namespace ns1{ int a = 3; }        //ns1이라는 이름 공간 정의
namespace ns2{ int a = 4; }        //ns2라는 이름 공간 정의
void main(){
  {
    int a = 5;
    cout<<a<<", "<<ns1::a<<", "<<ns2::a<<endl;  //출력 : 5, 3, 4
  }
  using namespace ns1;
  cout<<ns2::a<<", "<<a<<endl;                  //ns1::a, 출력 : 4, 3
}
```

// 이름이 명명되지 않은 이름 공간

- 이름 공간의 이름을 명명하지 않으면 변수는 전역변수와 비슷하게 사용된다.
- 이름이 없으므로 using을 사용하여 특정 영역에서 사용되도록 할 수는 없다.

```
#include <iostream>
using namespace std;
namespace { int a = 8; }   //이름이 없는 이름 공간
void main()
{
  cout<<a<<endl;           //출력 : 8
}
```

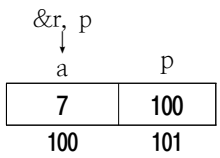
// 참조형과 포인터형

C++은 '참조형과 포인터형'을 모두 지원한다. Java는 참조형, C는 포인터형을 지원한다.

참조형	• 내부적 : 포인터처럼 적용되고 • 외부적 : 일반변수처럼 사용한다.

```
#include <iostream.h>
void main()
{
  int a = 7;            //일반변수 a
  int &r = a;           //참조형 r, 기호 &가 있으면 참조형 변수가 된다.
  int *p = &a;          //포인터형 p
  cout<<r<<"\n";        //출력 : 7
  cout<<*p<<"\n";       //출력 : 7
}
```

메모리 구조 : r과 p는 기억공간 a를 접근할 수 있다.



a = r = *p = 7

참조형 변수는 일반변수나 포인터처럼 새로운 기억공간에 할당되지 않는다.
이미 할당된 기억공간에 대한 다른 이름(변수 a의 별칭)으로 사용되어, 접근할 수 있을 뿐이다.

```
//참조형을 이용하여 두 변수의 값을 교환하는 프로그램
void swap(int& a, int& b){ int c = a; a = b; b = c; }
void main()
{
  int x = 6, y = 7;
  swap(x, y);   cout<<x<<"   "<<y<<endl;     //출력 : 7   6
}
```

// 인라인 함수(inline function; 확장함수)

인라인 함수는 부작용이 없으면서 처리속도를 빠르게 해 준다.

```
#include <iostream>
using namespace std;
#define hap1(a, b) a + b                          //매크로 함수
inline int hap2(int a, int b){ return a + b; }   //인라인 함수, inline이 있으면 인라인 함수
int hap3(int a, int b){ return a + b; }          //일반 함수
void main()
{
  int h1, h2, h3;
  h1 = 2 * hap1(3, 4);          //h1 = 2 * 3 + 4 = 10 (매크로 함수는 전처리기가 전개함)
  h2 = 2 * hap2(3, 4);          //h2 = 2 * 7 = 14 (인라인 함수는 컴파일러가 적절히 전개함)
  h3 = 2 * hap3(3, 4);          //h3 = 2 * 7 = 14 (일반 함수는 컴파일러가 처리함)
  cout<<h1<<", "<<h2<<", "<<h3<<endl;
}
```

실행 결과 : 10 14 14

매크로 함수	• 매크로 함수는 처리속도가 빠르다.(**단순 전개 방식**이므로) • 매크로 함수는 '인수형을 점검하지 않음' 등 여러 부분에서 **부작용을 발생**시킨다. • 매크로 함수는 **전처리기가 전개**한다.
일반 함수	• 일반 함수는 속도는 늦을 수 있으나 **부작용 발생은 없다.** • 일반 함수는 **컴파일러가 처리**한다.
인라인 함수	• 인라인 함수는 부작용이 없으면서 처리속도를 빠르게 해 준다. • 인라인 함수는 **컴파일러가 적절하게 전개**한다. • 인라인 함수에서는 재귀호출을 구현할 수 없다. • 함수 크기가 큰 것은 인라인 함수로 적당하지 않다. • 인라인 함수로 부적합 : goto, switch, for, while문 등이 사용된 복잡한 함수

• 인라인 함수와 일반함수의 차이점은 프로그래머가 원시코드를 어떻게 작성하느냐?에 있지 않고, C++ 컴파일러가 원시코드를 어떻게 적절히 처리하느냐?에 있다.
• 프로그래머가 인라인 함수로 정의하면 컴파일러는 반드시 따르는 것은 아니다.
• 즉, 프로그래머가 인라인 함수로 정의해도 함수 덩치가 크면 일반 함수로 컴파일 한다.
• 인라인 함수는 처리속도를 빠르게 할 목적으로 **간단한 문장**에 적용하는 것이 원칙이다.

기출문제 분석

1. 객체지향(object-oriented) 언어인 C++에 대한 설명으로 옳지 않은 것만을 <보기>에서 모두 고르면? [2020년 국회 9급]

----<보기>----------

ㄱ. 다중상속(multiple inheritance)이 불가능하다.

ㄴ. 함수 오버로딩(overloading)이 가능하나 한 클래스(class) 내에서 동일한 이름을 가진 다수의 함수는 정의될 수 없다.

ㄷ. 클래스를 통해 데이터 캡슐화(encapsulation)와 추상화(abstraction)를 지원한다.

ㄹ. 다형성(polymorphism)과 상속(inheritance)을 지원한다.

ㅁ. 다형성(polymorphism)의 지원을 위해 동적 바인딩(dynamic binding)을 사용한다.

---

① ㄱ, ㄴ ② ㄱ, ㄴ, ㅁ ③ ㄱ, ㄷ, ㄹ

④ ㄴ, ㄹ, ㅁ ⑤ ㄷ, ㄹ, ㅁ

♤ **객체지향언어 C++**

ㄱ. 다중상속(multiple inheritance)이 불가능하다.(×)

→ C++은 다중상속을 지원한다.

ㄴ. 함수 오버로딩(overloading)이 가능하나 한 클래스(class) 내에서 동일한 이름을 가진 다수의 함수는 정의될 수 없다.(×)

→ 한 클래스(class) 내에서 동일한 이름을 가진 다수의 함수가 정의될 수 있다.

정답 : ①

2. 다음 중 다중상속을 지원하는 언어를 모두 묶은 것은?

㉠ C++ ㉡ Python ㉢ C# ㉣ Java

---

① C++ ② C++, Python

③ C++, Python, C# ④ C++, Python, C#, Java

♤ **객체지향언어에서 다중상속**

• C++과 파이썬(python)은 **다중상속을 지원**한다. C#과 Java는 다중상속을 지원하지 않는다.

정답 : ②

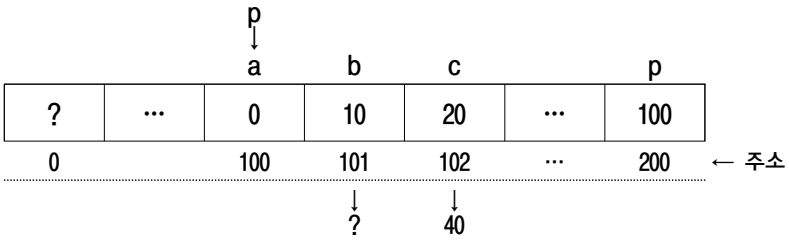
3. 다음은 C++ 프로그램이다. 의미 오류(semantic error)가 발생되는 문장은? [2009년 군무원 9급 유형]

```
-----------------------------------
#include <iostream>
using namespace std;
void main()
{
  int a = 0, b = 10, c = 20, *p;
  b = *a;
  c *= 2;
  p = &a;
  *p = c;
  cout<<*p<<endl;
}
-----------------------------------
```

① b = *a;　　② c *= 2;
③ p = &a;　　④ *p = c;

♤ 의미 오류(semantic error)

// 메모리 구조는 다음과 같다.



- b = *a;　　//오류 발생 (invalid type argument of unary)
  ↓
  ↓있는 그대로 설명하면
  ↓
- a = 0이므로 0번지의 내용이 변수 b에 대입된다.
- 메모리 0번지의 내용은 아무런 의미가 없다. 그 값이 무엇인지 알 수도 없다.

정답 : ①

4. 다음 C++ 프로그램의 실행 결과로 옳은 것은? [2019년 국회 9급]

```
------------------------------
#include <iostream>
using namespace std;
int main() {
  int x;
  for (x = 1; x <= 7; x++) {
      if (x == 5)
          continue;
      else if (x == 6)
          break;
      cout << x;
  }
  return 0;
}
------------------------------
```

① 123 ② 1234 ③ 12345
④ 12346 ⑤ 12347

♤ C++ 프로그램

```
int main() {
  int x;
  for (x = 1; x <= 7; x++)
  {
      if (x == 5)              //x == 5 이면 증감식으로
          continue;
      else if (x == 6)         //x == 6 이면 for문 강제 탈출
          break;
      cout << x;               //출력 : 1234
  }
  return 0;
}
```

정답 : ②