2008학년도 1학기

고급객체지향프로그래밍

중간 시험 모범 답안

강대기

문제 1) 다음은 템플릿을 이용하여 4의 팩토리얼(4!)을 구하는 프로그램이다.

```
#include <iostream>
template <int N> class Factorial { public: enum { value = N * Factorial<N - 1>::value }; }; // 템플릿
template <> class Factorial<0> { public: enum { value = 1 }; }; // 템플릿의 특수화
int main()
{
  std::cout << Factorial<4>::value << std::endl;
  return 0;
}
```

(a) 위의 프로그램을 참고하여, 클래스 템플리트를 이용하여 피보나치 급수 10 항(즉 Fib(10))을 구하는 프로그램을 작성하라.
처음 항과 둘째 항인 Fib(1) = 1 이고 Fib(2) = 1 로 정한다.

```
#include <iostream>

template <int N> struct Fibo
{ enum { value = Fibo<N-1>::value +  Fibo<N-2>::value }; };

template <> struct Fibo<2> { enum { value = 1 }; };
template <> struct Fibo<1> { enum { value = 1 }; };

int main()
{
    std::cout << Fibo<10>::value << std::endl;
    return 0;
}
```

(b) 클래스 템플리트를 이용하여 4 개의 디스크와 3 개의 말뚝을 가진 하노이의 탑을 푸는 프로그램을 작성하라.
단순히 자료형을 템플리트로 하지 말고, 위의 예처럼 템플리트 메타 프로그래밍을 사용해야 한다.

하노이의 탑을 모르는 사람을 위해 설명하자면, 퍼즐의 일종이다. 세 개의 기둥과 이 기둥에 꽂을 수 있는 크기가 다양한 원판이 있고, 퍼즐을 시작하기 전에는 한 기둥에 원판들이 작은 것이 위에 있도록 순서대로 쌓여 있다.
게임의 목적은 다음 두 가지 조건을 만족시키면서, 한 기둥에 꽂힌 원판들을 그 순서 그대로 다른 기둥으로 옮겨서 다시 쌓는 것이다.

1. 한번에 하나의 원판만 옮길 수 있다.
2. 큰 원판이 작은 원판위에 있어서는 안된다.

```
#include <iostream>
using namespace std;
template <int From, int To, int Using, int N> class Hanoi {
public:  static inline void move() {
                    Hanoi<From, Using, To, N-1>::move();
                    cout << From << "->" << To << endl;
                    Hanoi<Using, To, From, N-1>::move();
          };
};

template <int From, int To, int Using> class Hanoi<From, To, Using, 1>
{ public: static inline void move() { cout << From << "->" << To << endl;};   };

int main()
{
  Hanoi<1,2,3,4>::move();
  return 0;
}
```

문제 2) 다음에 대해 한 문장으로 서술하라.

(a) 기초 클래스
두 개의 클래스가 상속 관계일 때, 상속되는/추상화된 클래스이다.

(b) 파생 클래스
두 개의 클래스가 상속 관계일 때, 상속받는/확장되는 클래스이다.

(c) 추상 기초 클래스
순수 가상 함수(f 참고)를 하나 이상 가지는 클래스를 추상 클래스라고 하는 데, 상속 관계에서 기초 클래스가 추상 클래스인 경우를 추상 기초 클래스라고 한다.

(d) 가상 기초 클래스
다중 상속으로 인해 다이아몬드 상속 관계가 되면, 최종적으로 파생된 클래스 안에는 하나의 조상 클래스 객체가 여러 카피가 존재하여, 멤버의 중복으로 인한 메모리 낭비와 모호함이 발생하는 데, 이 문제를 해결하기 위해 특정 기초 클래스에 대해 간접적으로 여러 번 상속되더라도 한번만 상속시키게 지정된 클래스를 가상 기초 클래스라고 한다.

(e) 가상 함수
가상 함수는 클래스의 멤버 함수인데, 호출될 때 가상 테이블 (vtab)에 의해 간접적으로 주소가 지정되는 함수로, 더 설명하자면 비가상 함수는 호출되면 컴파일 시간에 그 주소가 계산(early binding)되어 들어가는 반면, 가상 함수는 호출되면 실행 시간에 그 클래스의 가상 테이블에 적힌 주소가 계산(late binding)됨.
(따라서, 예를 들어, 파생 클래스 객체가 기초 클래스의 포인터나 참조로 업캐스팅되어 어드레싱되는 경우, 멤버 함수를 호출하면 가상 테이블을 통해 그 파생 클래스의 가상 멤버 함수가 호출됨)

(f) 순수 가상 함수
기초 클래스 내부의 가상 함수를 선언할 때, 함수 원형 뒤에 “=0”을 붙이면 순수 가상 함수가 되는 데, 기초 클래스의 순수 가상 함수는 파생 클래스에서는 반드시 재정의해야 함.

(g) 비가상 함수 (가상 함수가 아닌 멤버 함수)
비가상 함수는 클래스의 멤버 함수로, 호출되면 컴파일 시간에 그 주소가 계산(early binding)되어 들어감. (예를 들어, 파생 클래스 객체가 기초 클래스의 포인터나 참조로 업캐스팅되어 어드레싱되는 경우, 멤버 함수를 호출하면 기초 클래스의 비가상 멤버 함수가 호출됨.)

(h) C++에서 struct와 class의 차이는 무엇인가?

struct 는 멤버에 대한 default 접근 제어가 public이고 class는 private이다.

문제 3) 정수 배열(예: int arr[100])이 두 개가 있다. 두 개의 배열은 작은 수부터 큰 수로 정렬되어 있고, 두 배열의 크기는 같을 수도 있고 서로 다를 수도 있다. 두 개의 배열에 공통으로 들어가 있는 정수들만 출력하는 프로그램을 작성하라. 예를 들어 배열 하나가 1,2,3,4,5,11 이고 다른 하나가 2,5,7,8,11,14,20 이면, 2,5,11 이 출력될 것이다. 이 프로그램은 객체 지향적으로 작성할 필요는 없다.

```
#include <iostream>
using std::cout;
using std::endl;
void printCommonVals(int* ar1, int ar1len, int* ar2, int ar2len)
{
  bool done=false;
  int index1=0, index2=0;
  while (!done)  {
    if (ar1[index1]==ar2[index2]) {
      cout << ar1[index1] << " ";
      index1++; index2++;
    }
    else if (ar1[index1]>ar2[index2]) index2++;
    else index1++;
    if (index1>=ar1len) done=true;
    if (index2>=ar2len) done=true;
  }
  cout << endl;
}

int main()
{
  int ar1[6] = {1,2,3,4,5,11};
  int ar2[7] = {2,5,7,8,11,14,20};
  printCommonVals(ar1,6,ar2,7);
  return 0;
}
```

문제 4) 다음 프로그램의 출력을 적어라.

```
#include <iostream>
using std::cout;
using std::endl;
template <typename T>
class beta
{
private: template <typename V>            // 내포된 템플릿 클래스 멤버
         class hold
         {
         private: V val;
         public:  hold(V v = 0) : val(v) { }
                  void show() const { cout << val << endl; }
                  V Value() const { return val; }
         };
         hold<T> q;                                    // 템플릿 객체
         hold<int> n;                                  // 템플릿 객체
public:
         beta(T t, int i) : q(t), n(i) { }
         template <typename U>            // 템플릿 메서드
         U blab(U u, T t) { return (2 * n.Value() +  3 * q.Value()) * u / t; }
         void Show() const { q.show(); n.show(); }
};

int main()
{
         beta<double> guy(2.0, 3);
         cout << guy.blab(10, 50) << endl;
         cout << guy.blab(10., 50) << endl;
         return 0;
}
```

2

2.4

문제 5) 이를테면 휴대폰같이 메모리가 극히 부족한 상황에서 돌아가는 프로그램을 작성해야 한다고 고려해 보자.
무선 인터넷을 통해 1부터 100까지의 정수들 중 99 개의 정수들이 무작위(random)로 입력되어 들어오되, 한번 들어온 숫자는 다시 들어오지 않는다. 이렇게 99 개의 정수를 받아들이고 나서 빠진 하나의 정수를 찾는 프로그램을 작성하라.
단, 메모리가 지극히 부족하여 배열이나 그런 비슷한 것은 절대 사용할 수 없으며, 이 휴대폰에서 int 형은 2 바이트이다.
객체지향적으로 프로그램을 작성할 필요는 없다.

```
int sum=5050;
for (int i=0;i<99;i++) sum-=getTheNextNum();
cout << sum << endl;
```

문제 6) 다음 질문에 간결히 답하시오.

(a) 클래스의 이름이 Student 일 때, 복사 생성자의 선언을 적어라. 복사 생성자의 매개 변수에 참조가 들어가는 이유는 무엇인가?

Student(const Student&);

복사 생성자의 매개 변수에 참조가 들어가지 않고 그냥 객체로 선언되면 복사 생성자를 호출하면서 매개 변수 간의 복사가 일어날 때 복사 생성자가 다시 호출되므로, 무한 루프를 돌게 됨.

(b) 가상 기초 클래스를 사용할 경우, 파생 클래스의 생성자에서 명시적으로 조상 클래스의 생성자도 호출해 줘야 하는 이유는 무엇인가?

비가상으로 다중 상속을 하게 되면, 각각의 기초 클래스가 자신들 만의 조상 클래스 객체를 가지고 있으며 각각의 생성자에서 조상 클래스의 생성자를 호출할 수 있으나, 가상으로 다중 상속을 하게 되면 조상 클래스 객체는 하나 뿐이고, 어떤 기초 클래스에서 조상 클래스 객체의 생성자를 호출해야 할지 애매모호하게 되므로, 파생 클래스에서도 명시적으로 조상 클래스의 생성자를 호출해야 함.

(c) 가상 기초 클래스의 경우 비교 우위(dominance)가 생기는 이유는 무엇인가?

가상으로 다중 상속을 받게 되면, 조상 클래스 객체가 단 하나이므로 실질적인 상속 그래프(inheritance)를 그릴 수 있다. 이 상속 그래프에서 위상적으로 더 높은 위치에 있는 이름에 대해 우위를 줄 수 있다.

(d) 상속을 해야 할 경우, 파괴자를 가상으로 선언하는 이유는 무엇인가?

파괴자가 가상 함수로 선언되어야 그 클래스 객체가 파괴될 때, 그 객체에 맞는 파괴자 함수가 호출된다.

예를 들어, 파생 클래스 B가 기초 클래스 A에게서 확장되었고 B 객체를 A 포인터로 받았을 때, 파괴자들이 가상으로 선언되어 있지 않으면, A 포인터에 대해 delete를 하면 B 객체임에도 A의 파괴자가 호출된다. 이러한 문제를 막기 위해 가상으로 선언해야 한다.

(e) 선언하지 않아도 컴파일러가 몰래 만드는 멤버 함수들은 어떤 것들이 있는가?

선언된 생성자가 없을 경우엔 디폴트 생성자, 복사 생성자, 복사 대입 연산자, 파괴자

그 외 주소 연산자 (operator&), 콤마 연산자 (operator,)

문제 7) 링크드 리스트의 내용들을 반전시키는 프로그램을 작성하라. 링크드 리스트의 각 노드들은 정수 값을 가지고 있다.
예를 들어 링크드 리스트의 내용이 1-> 2 -> 100 -> 50 -> 37 -> 237 이면, 프로그램을 수행한 결과는 237 -> 37 -> 50 -> 100 -> 2 -> 1 로 바뀌어야 한다. 객체지향적으로 프로그램을 작성할 필요는 없다. (미국 마이크로소프트 면접 문제)

```
Node *revList = NULL;
Node *currNode = currHead;

while (NULL != currNode)
{
  currHead=currNode->next;
  currNode->next=revList;
  revList=currNode;
  currNode=currHead;
  currHead=revList;
}
```