객체지향프로그래밍 숙제 #2 해답

강대기

2007년 10월 25일

제 1 절 연습 문제 숙제

2.1 C++ 프로그램을 구성하는 모듈은 함수이다.

구조체나 클래스는 그 안에 다른 구조체나 클래스를 정의할 수 있는 반면, 원칙적으로 함수는 중첩된 정의가 불가능하다. 즉 함수 안에서 다른 함수를 정의할 수 없다는 의미이다. 예를 들어 다음과 같은 코드는 틀린 것이다. C의 89년 표준안인 C89에서도 틀리고, 99년 표준안인 C99에서도 틀린 것으로 간주되며, C++에서도 틀린 것이다.

```
#include <stdio.h>
void func1()
{
  printf("func1\n");
  void func2()
  {
    printf("func2\n");
  }
  func2();
}

int main()
{
  func1();
  return 0;
}
```

다만 GNU GCC 같은 일부 컴파일러들은 저런 식의 함수 내 함수 정의를 허용한다.

2.2 `#include` 지시어는 그 다음에 오는 헤더 파일의 내용을 그 위치에 그대로 삽입한다. 따라서 `iostream` 이라는 헤더 파일의 내용을 `#include` 가 있는 위치에 그대로 삽입한다.

더불어 C++ 에서는 C 에서 사용하던 헤더 파일의 확장자인 "`.h`"를 사용하지 않는다.

2.3 std 이름 공간에 정의되어 있는 이름들을 프로그램에서 사용할 수 있게 허용한다.

이름 공간에 대해서는 좀 더 본격적으로 9장에서 배우게 된다. using 선언(declaration)과 using 지시자(directive)가 있으며 각각 다음과 같이 사용된다.

using 선언은 하나의 이름만 그 공간에서 사용할 수 있게 한다. 만일 이미 같은 이름이 있다면 충돌이 일어나 컴파일러가 불평을 한다.

```
using std::cin;
using std::cout;
using std::cerr;
using std::clog;
using std::endl;
```

반면, using 지시자는 그 이름 공간 안의 모든 이름들을, 지시자가 있는 공간에서 사용할 수 있게 한다. 만일 이미 같은 이름이 있다면, 나중에 선언된 것이 전에 선언된 것을 가린다(override).

```
using namespace std;
```

2.9 froop 함수는 double 형인 t 를 하나의 전달 인자로 받고, 정수 값을 반환한다. 예를 들어 다음과 같이 사용된다.

```
int val = froop(3.14);
```

rattle 함수는 int 형인 n 를 하나의 전달 인자로 받고, 아무 것도 반환하지 않는다. 예를 들어 다음과 같이 사용된다.

```
rattle(3);
```

prune 함수는 아무 전달 인자도 받지 않고, 정수 값을 반환한다. 예를 들어 다음과 같이 사용된다.

```
int val = prune();
```

2.10 함수의 반환 형이 void 이면 return 키워드를 사용할 필요가 없다. 함수 내에서 실행을 마치고 원래 호출한 함수로 돌아가고 싶을 때 그 위치에서 return 키워드를 사용하면 된다.

교재에선, 리턴 값을 제공하지 않는 경우 return; 이라고 사용할 수 있다고 했으나, C++ 와 C99 에서는 금지되었다.

3.1 크기가 다른 정수형이 여러 개 제공되므로, 프로그래밍을 할 때 요구되는 상황에 맞춰서 선택할 수 있는 폭이 넓다. short를 사용하면 메모리를 절약할 수 있고 (이른바 memory padding 때문에 실제로는 그렇지 않을 수도 있지만...), long을 사용하면 대부분의 경우에 대한 충분한 저장 용량이 확보된다.

3.2
```
short a = 80;
unsigned int b = 42110;
int c = 3000000000;
```

3.3 C++는 정수값의 한계를 벗어나지 않도록 사용자를 보호하는 안전 장치를 제공하지 않는다. 이러한 한계들은 시스템에 따라 다르며 `climits` 헤더 파일을 참조해야 한다.

3.5 대부분의 시스템이 기본적으로 ASCII 코드를 사용하므로, 두 명령문은 대부분 같은 결과를 만들어내지만, 예를 들어 EBCDIC 코드를 사용하는 경우에는 다른 결과를 내게 된다.

첫번째 문장은 `grade`에 `int` 형 상수 65를 저장하므로 ASCII 코드를 사용하는 시스템에서는 'A'가 저장된 효과를 낸다. 두번째 문장은 `grade`에 `char` 형 상수 'A'를 저장하므로 어떠한 시스템에서도 'A'가 저장된다.

3.6
```
char c = 88;
cout << c;

cout.put(char(88));

cout << char(88);

cout << (char)88;
```

3.7 두 데이터 형 중 어느 것이 더 크냐 또는 같은가에 달려있다. `long` 형이 4 바이트를 차지하면 10자리 숫자까지 저장할 수 있는 데, `double` 형은 유효숫자를 13 자리까지 허용하므로, 이런 경우 데이터 손실은 없다.

4.1
a. `char actors[30];`
b. `short betsie[100];`
c. `float chuck[13];`
d. `long double dipsea[64];`

4.5
```
char lunch[] = "cheeseburger";
```

4.6
```
struct fish
{
  char kind[20];
  int weight;
  float length;
};
```

4.12 올바르다. 문자열 상수는 그 시작 주소로 평가되므로, `(int*)`에서 보듯이 char가 아닌 `int` 같은 다른 형(type) 에 대한 포인터로 타이프캐스팅(type casting)을 하는 경우, `cout` 객체는 `(char*)` 에 대해 수행하는 것과는 다른 동작을 한다.

(char*) 형으로 지정된 경우에는 그 주소를 시작 번지로 하는 문자열을 출력하는 반면, 그 외의 경우에는 그 주소 자체를 unsigned 형으로 출력한다. 이에 대해서는 다음의 코드를 실행해 보면 이해가 빠를 것이다.

```
cout << (char*)"Hello, World!" << endl;
cout << (unsigned char*)"Hello, World!" << endl;
cout << (signed char*)"Hello, World!" << endl;

cout << (void*)"Hello, World!" << endl;
cout << (int*)"Hello, World!" << endl;
cout << (short*)"Hello, World!" << endl;
cout << (long*)"Hello, World!" << endl;
cout << (long long*)"Hello, World!" << endl;
cout << (float*)"Hello, World!" << endl;
cout << (double*)"Hello, World!" << endl;
cout << (long double*)"Hello, World!" << endl;
cout << (bool*)"Hello, World!" << endl;
```

4.13
```
struct fish
{
  char kind[20];
  int weight;
  float length;
};

fish * pole = new fish;
cout << "물고기의 종류를 입력하십시오:";
cin.getline(pole->kind,20);
delete pole;
```

제 2 절 프로그래밍 문제 숙제

2.4 ex02-04.cpp

2.5 ex02-05.cpp

3.2 ex03-02.cpp

3.3 ex03-03.cpp

3.4 ex03-04.cpp

3.6 ex03-06.cpp

4.1 ex04-01.cpp

4.2 ex04-02.cpp

4.6 `ex04-06.cpp`

4.8 `ex04-08.cpp`