객체지향프로그래밍 숙제 #3

강대기

2007년 9월 28일

제 1 절 숙제 제출에 대해 반드시 알아야 할 사항

기본적으로 숙제는 내준 시점에서 2주 뒤까지 제출한다. 본 숙제는 세번째 주(9/17—9/21)에 내주었으므로 2주 뒤인 10/1—10/5 에 제출해야 할 것이다. 그러나 E11반과 E13 반은 추석이 겹쳐있고, 또한 E13 반의 경우 개천절이 겹쳐있다.

그러므로 E11, E12, E13 세 반 모두 일괄적으로 10월 12일까지 제출해야 하도록 한다.

숙제를 늦게 제출할 경우, 하루에 20% 씩 점수가 깍인다. 이를테면, 100점 만점에서 하루 늦는 경우는 만점을 맞아도 80 점이 된다. 이틀 늦는 경우는 만점을 맞아도 60점이다.

숙제를 제출할 때는 연습 문제를 푼 내용들을 하나의 텍스트 파일로 작성하여 이메일 **dkkang@dongseo.ac.kr**로 보낸다. 프로그램 숙제들은 실행 파일은 보내지 말고, 소스 코드만 작성하여 전체를 하나의 ZIP 파일로 묶어서 보낸다. 따라서, 보낼 파일의 개수는 연습문제를 풀은 텍스트 파일 하나와 프로그램 소스들을 묶은 ZIP 파일 하나로 두 개가 된다.

소스 코드의 파일 이름은, 별도로 지정하지 않는 한, 프로그래밍 문제의 번호를 토대로 만든다. 예를 들어 프로그래밍 문제 5.1의 경우, `5_1.cpp` 라고 이름을 지으면 된다.

한글 인코딩은 되도록이면 EUC-KR을 코딩을 따르되, UTF-8도 허용가능하다. 이메일로 제출할 때, 제목을 "[OOP2007]"로 시작하고 "HW3", 반, 학번, 이름 등을 순서대로 기재한다. 각 필드는 '.'로 끊어준다.

예를 들어 E11반 학번 20061234인 홍길동이 숙제 #3을 제출하는 경우 다음과 같이 메일 제목을 쓰면 된다.

```
제목: [OOP2007]HW3.E11.20061234.홍 길 동
```

제 2 절 숙제

2.1 읽기 숙제 (제출할 필요 없음)

"C++기초플러스 - 5판"의 5장과 6장

2.2 풀어보기 숙제 (제출할 필요 없음)

"C++기초플러스 - 5판"의 5장과 6장의 모든 연습 문제들

2.3 연습 문제 숙제 (텍스트 파일로 제출)

다음의 문제들을 풀어서 텍스트 파일로 제출한다.

1. 연습문제 5.1
2. 연습문제 5.9
3. 연습문제 6.1
4. 연습문제 6.2
5. 연습문제 6.3
6. 연습문제 6.9
7. Duff's device 에 대해 한 페이지 분량으로 서술한다. (난이도 下)
8. 세미콜론(';')을 사용하지 않고 "Hello World"를 출력하는 프로그램을 작성한다. 즉 "Hello World"를 출력하는 프로그램을 짜되, 그 프로그램 내에는 세미콜론(';')이 없어야 한다. (난이도 中)

2.3.1 중복되지 않는 난수들 (난이도 中)

아홉번째 문제는 다음과 같다.
우선 다음의 프로그램을 보자.

```
#include <iostream>
#include <ctime>

int main ()
{
  using namespace std;
  srand((unsigned)time(NULL));
  int d=rand()%12;
  cout << d << endl;
  return 0;
}
```

이 프로그램은 실행할 때마다 0에서 11 사이의 정수값 (0과 11 포함) 중 하나를 무작위로 출력한다. 이 점은 다음을 실행해 보면 더 확실히 알 수 있다.

```
#include <iostream>
#include <ctime>

int main ()
```

```
{
  using namespace std;
  srand((unsigned)time(NULL));

  for (int i=0;i<100;i++)
  {
    int d=rand()%12;
    cout << d << " ";
  }
  cout << endl;
  return 0;
}
```

위의 프로그램은 실행할 때마다 다른 값들을 출력한다. 0에서 11까지의 값에서 난수를 100번 출력하므로 어떤 값은 중복될 수도 있음을 알 수 있다.

이러한 rand 함수를 난수 발생 함수라고 한다. 근본적으로 컴퓨터는 결정적이므로, 진정한 난수 발생 함수는 불가능하며, 컴퓨터로 구현되는 난수 발생 함수는 초기값을 필요로 한다. 위의 srand는 이러한 난수 발생 함수의 초기값을 그 프로그램이 실행하는 당시의 시간값으로 초기화하는 함수이다.

rand 함수는 0에서 RAND_MAX-1 범위에서 난수를 발생시킨다. (0과 RAND_MAX-1 도 포함...) RAND_MAX 의 값을 알고 싶으면 다음과 같이 프로그램을 짜서 실행해 보면 알 수 있다.

```
#include <iostream>

int main ()
{
  std::cout << RAND_MAX <<std::endl;
  return 0;
}
```

이제 다음의 프로그램을 보자. 다음의 프로그램은 0에서 32760 사이의 (0과 32760 포함) 32760개의 난수를 발생시킨다.

```
#include <iostream>
#include <ctime>

int main ()
{
  using namespace std;
  srand((unsigned)time(NULL));

  for (int i=0;i<32760;i++)
  {
    int d=rand()%32761;
    cout << d << " ";
  }
```

```
  cout << endl << "The end!\n";
  return 0;
}
```

문제는 이 경우에도 발생되는 32760 개의 난수들 중에는 이미 앞에서 출력되어 나온 난수가 다시 나올 수 있다는 것이다. 참고로 0에서 32760 사이(0과 32760 포함)의 정수는 전부 32761 개가 있다. (왜냐면 0을 포함하므로...)

이제 본격적으로 문제를 내겠다.

이제 30000 개의 난수를 발생시키되, 출력되는 어떠한 값도 이미 나온 값이 다시 나오지 않도록 하는 프로그램을 작성하라. 즉, 중복되지 않는 30000 개의 서로 다른 0에서 32760 사이(0과 32760 포함)의 난수를 빠르게 출력하는 프로그램을 작성하라는 것이다.

가장 직설적이고 멍청한 해답은 32761 개의 bool 값을 가지는 배열을 만들어서 난수가 나올 때마다 해당 값을 true로 설정하고 이미 나온 경우 다시 rand 함수를 돌리는 방법일 것이다. 그러나 이러한 경우, 뒤로 갈수록 프로그램의 실행은 점점 느려지면서 난수값이 천천히 나오게 된다. (요즘 컴퓨터는 저 정도는 충분히 커버할 수 있을 정도로 빠르므로 못 느낄 수 있지만...)

물론 모든 난수가 동일한 속도로 빠르게 나오는, 그러나 중복되는 난수는 없는, 뛰어난 해법이 존재한다. 직설적이고 멍청한 솔루션도 점수를 주겠으나, 더 빠른 해법으로 프로그램을 만들어 제시하는 경우, 보너스 점수로 10점을 더 주도록 하겠다.

본 문제는 실제로 SI 업체나 소프트웨어 개발 회사의 입사 면접에서 사용된 문제이다. 직설적이고 멍청한 해법은 왠만한 프로그래머라면 다 제시할 수 있으며, 더 빠를 해법을 제시하는 경우에만 주로 면접을 합격하고 취업이 된다.

2.3.2 중첩된 /* */ 커멘트 (난이도 上)

열번째 문제는 다음과 같다.

기본적으로 C에서는 /* */를 주석(커멘트; comment)로 사용한다. 다음의 프로그램 코드를 보자.

```
#include <iostream>
int main()
{
  using namespace std;

  std::cout << "Hello, the Universe 1\n";
  std::cout << "Hello, the Universe 2\n";
  std::cout << "Hello, world\n"; /* Print Hello world */
  std::cout << "Hello, the Universe 3\n";
  std::cout << "Hello, the Universe 4\n";

  std::cout << "Hello, Dongseo Univ.\n";
  return 0;
}
```

여기서 “Hello, the Universe”로 시작하는 문자열들과 “Hello, world”를 출력하는 부분을 일시적으로 컴파일되어 실행되지 않도록 하기 위해 프로그래머가 /* */로 둘러싼 다음 경우를 보자.

```
#include <iostream>
int main()
{
  using namespace std;

/*
  std::cout << "Hello, the Universe 1\n";
  std::cout << "Hello, the Universe 2\n";
  std::cout << "Hello, world\n"; /* Print Hello world */
  std::cout << "Hello, the Universe 3\n";
  std::cout << "Hello, the Universe 4\n";
*/

  std::cout << "Hello, Dongseo Univ.\n";
  return 0;
}
```

이 프로그램은 컴파일이 되지 않고 에러가 난다. 그 이유는 중첩된 커멘트(nested comments)를 대부분의 C++ 컴파일러가 허용하지 않기 때문이다. 중첩된 커멘트란, /* */ 로 이루어지는 커멘트 블럭 내에서 다시 /* */를 사용할 수 있는 경우이다. 중첩된 커멘트가 사용 가능하다면, 위의 프로그램이 에러없이 컴파일 되고, 실행되면 “Hello, Dongseo Univ.\n”만 출력될 것이다.

따라서, 위와 같이 특정 부분의 코드를 일시적으로 컴파일되지 않도록 막고 싶다면 ‘#ifdef false’ 와 ‘#endif’ 를 사용하는 게 정석이다. (C 언어의 경우에는 ‘#ifdef false’ 대신 ‘#ifdef 0’ 를 사용한다.)

```
#include <iostream>
int main()
{
  using namespace std;

#ifdef false
  std::cout << "Hello, the Universe 1\n";
  std::cout << "Hello, the Universe 2\n";
  std::cout << "Hello, world\n"; /* Print Hello world */
  std::cout << "Hello, the Universe 3\n";
  std::cout << "Hello, the Universe 4\n";
#endif

  std::cout << "Hello, Dongseo Univ.\n";
  return 0;
}
```

이제 배경 지식을 설명하였으므로, 본격적으로 문제를 내도록 하겠다. 대부분의 C/C++ 컴파일러가 중첩된 커멘트를 허용하지 않는 반면, 볼랜드 C/C++ 컴파일러(Borland C/C++ Compiler)의 경우, 중첩된 커멘트를 사용가능하다.

하나의 C++ 프로그램을 작성하는 데, 중첩된 커멘트를 허용하는 컴파일러(예를 들어 볼랜드 컴파일러)로 컴파일하면 "This compiler accepts nested comments"라고 출력하고(이 컴파일러는 중첩된 커멘트를 허용합니다라는 뜻), 중첩된 커멘트를 허용하지 않는 컴파일러로 컴파일하면 "This compiler does not accept nested comments"라고 출력하도록 한다(이 컴파일러는 중첩된 커멘트를 허용하지 않습니다라는 뜻). 물론 이 C++ 프로그램은 중첩된 컴파일러를 허용하는 컴파일러로도 에러 없이 컴파일되고, 중첩된 컴파일러를 허용하지 않는 컴파일러로도 에러 없이 컴파일되야 한다.

2.4 프로그래밍 문제 숙제 (소스 파일들을 하나의 ZIP으로 제출)

프로그램들의 점수는 5점씩이며, 6개이므로 30점이다.

- 5.3, 5.6, 5.8, 5.9, 6.4, 6.7

그리고 다음의 세가지 프로그래밍 문제를 풀어서 제출한다.

2.4.1 숫자 알아맞추기 게임

이 문제의 점수는 20점이며, 난이도는 下이다.

위에서 우리는 srand 함수와 rand 함수에 대해 논하였다. 이제 짜야 할 프로그램은 다음과 같다.

컴퓨터와 당신은 숫자 알아맞추기 게임을 한다. 컴퓨터가 1에서 100 사이의 (1과 100 포함) 임의의 수 하나를 생각해 낸다. 컴퓨터는 당신에게 그 수를 알려주지 않고, 대신 당신은 컴퓨터에게 임의의 숫자를 입력한다. 당신이 입력한 그 수가 컴퓨터가 생각한 수이면, 컴퓨터는 "Correct!"라고 출력하고 종료한다. 만일 그 수가 컴퓨터가 생각한 수보다 크다면, "Too high." 라고 출력하고 다시 당신의 입력을 기다린다. 만일 그 수가 컴퓨터가 생각한 수보다 작다면, 'Too low." 라고 출력하고 다시 당신의 입력을 기다린다.

프로그램의 실행 예는 다음과 같다.

```
I have thought of one number between 1 and 100!
Enter your guess: 50
Too high!
Enter your guess: 25
Too high!
Enter your guess: 12
Too low!
Enter your guess: 18
Too low!
Enter your guess: 21
Too high!
Enter your guess: 20
```

```
Correct!
The end.
```

이 예에서 처음에 컴퓨터가 추측한 값은 20이다. 물론 이 값은 난수 발생 함수로 만들어야 한다.

프로그램의 이름은 Number.cpp 로 한다.

2.4.2 Strike/Ball 야구 게임

이 문제의 점수는 20점이며, 난이도는 下이다.

기본적으로 Strike/Ball 야구 게임은 한 플레이어가 1에서 9까지 겹치지 않는 숫자 3개를 추측하면, 다른 플레이어가 순서에 맞춰서 숫자를 10회 안에 맞추는 게임이다.

여기선 컴퓨터가 1에서 9까지의 9개의 숫자 중 차례로 3개를 고른다. (물론 화면에 출력하진 않는다.) 그러면 사용자가 이 3 개의 숫자를 추측하여 입력한다.

컴퓨터는 입력된 추측에 대해

- 숫자와 자릿수가 맞는 것은 Strike
- 숫자를 맞지만 자릿수가 맞지 않으면 Ball
- 숫자가 맞은 것이 하나도 없으면 3 Out
- 숫자와 자릿수가 세 개 다 맞으면 3 Strikes로 삼진 아웃으로 종료

으로 표기하고, 10회가 지나면 종료한다.

실행 예는 다음과 같다.

```
Okay! Go!
1. enter your guess: 1 2 3
0 Strike, 0 Ball
2. enter your guess: 4 5 6
0 Strike, 0 Ball
3. enter your guess: 7 8 9
1 Strike, 3 Ball
4. enter your guess: 8 9 7
1 Strike, 3 Ball
5. enter your guess: 9 7 8
1 Strike, 3 Ball
6. enter your guess: 9 8 7
0 Strike, 3 Ball
7. enter your guess: 8 9 7
1 Strike, 3 Ball
8. enter your guess: 7 9 8
3 Strike, 3 Ball
Congratulations!
```

위의 예에서는 컴퓨터가 처음에 7,9,8을 선택한 경우이다. 물론 컴퓨터는 서로 다른 3 개의 1과 9 사이의 어떤 값들이건 선택할 수 있다.

프로그램의 이름은 Baseball.cpp 로 한다.

2.4.3 Ones

이 문제의 점수는 30점이며, 난이도는 上이다. 이 문제는 ACM ICPC 프로그래밍 컨테스트를 위한 연습 문제 중 하나이다.

0과 10000 사이의 정수 n이 있다고 하자. ($0 \leq n \leq 10000$). 이 정수는 2로 나누어 떨어지지 않으며, 5로도 나누어 떨어지지 않는다. (즉 2로 나눈 나머지가 0이 아니며, 5로 나눈 나머지도 0이 아니다.)

이 정수 n 에다가 어떤 다른 정수를 곱하면 그 결과는 전부 1일 숫자가 나온다. 2와 5로 나누어지지 않는 0과 10000 사이의 모든 정수에 대해, 곱해서 전부 1인 수가 나오는 다른 정수가 분명히 있다.

예를 들어 9는 2로 나누어 떨어지지 않으며, 5로도 나누어 떨어지지 않는다. 그런데 9에 12345679를 곱하면 111111111 이 나온다. 즉 전부 1인 숫자가 나오는 것이다. 여기서 1의 갯수는 9개임을 알 수 있다.

또한 3의 경우를 보면 3 * 37 = 111 로, 3에 37을 곱하면 111이 되고, 1의 개수는 3이다. 그리고 7의 경우를 보면, 7 * 15873 = 111111 로, 7에 15783을 곱하면 111111이 되고, 1의 개수는 6이다. 9901의 경우를 보면, 9901 * 11222211 = 111111111111 로, 9901에 11222211 을 곱하면 111111111111이 되고, 1의 개수는 12 개이다.

이제 정식으로 문제를 내겠다.

2와 5로 나누어 떨어지지 않는 0과 10000 사이의 특정한 정수 n ($0 \leq n \leq 10000$) 에 대해 어떤 정수를 곱하면 이렇게 전부 1인 숫자가 나오는 여러 경우들 중 가장 작은 숫자에 대해 1의 갯수를 구하는 프로그램을 작성하라.

프로그램의 이름은 Ones.cpp라고 한다.

실행 예는 다음과 같으며, -1을 입력하면 종료한다.

```
Enter number: 2
2 is divisible by 2.
Enter number: 4
4 is divisible by 2.
Enter number: 5
5 is divisible by 5.
Enter number: 9
The number of 1's is 9.
Enter number: 3
The number of 1's is 3.
Enter number: 7
The number of 1's is 6.
Enter number: 9901
The number of 1's is 12.
Enter number: 9981
The number of 1's is 9972.
Enter number: 9011
```

```
The number of 1's is 9010.
Enter number: 9029
The number of 1's is 9028.
Enter number: 9059
The number of 1's is 9058.
Enter number: 9069
The number of 1's is 9066.
Enter number: 1
The number of 1's is 1.
Enter number: -1
The end!
```

이 프로그램에서 주의할 점은 기존의 short, int, 또는 long의 범위를 넘은 큰 수에 대해서도 결과가 제대로 나와야 한다는 거이다. 예를 들어 19의 경우를 보면, $19 * 5847953216374269 = 111111111111111111$ 이므로 1 의 개수는 18개나 된다. 29의 경우에는 $29 * 38314176245210727969348659 = 1111111111111111111111111111111$ 로, 1의 개수는 28개이다. 9981 인 경우에는 1의 개수는 9972 나 된다. C++에서 가장 큰 변수형의 유효숫자도 간단하게 넘어선다는 점을 알고 그 점을 고려해서 프로그래밍해야 한다.