고급객체지향프로그래밍 과제 #3

강대기

2008년 5월 14일

과제 제출에 대해 반드시 알아야 할 사항

본 과제의 데드라인은 5월 25일 일요일 밤 11시 59분이다.

본 과제는 13 문제이다. 앞의 문제들은 객체 지향에 대한 지식을 테스트하고, 뒤의 문제들은 기초적인 알고리즘에 대한 구현 능력을 테스트한다.

모든 과제 관련 정책은 과제 #2와 동일하다.

1. 다음과 같이 템플리트 메타 프로그래밍을 이용하여 5 개의 디스크와 3 개의 말뚝을 가진 하노이의 탑을 푸는 메타 함수를 작성하라.

   좀 더 설명하자면, 이를 위해 클래스 템플리트를 작성해야 한다. 또한 템플리트 메타 프로그래밍이므로 반드시 메타 함수를 작성해서 하노이의 탑을 구현해야 한다. 메타 함수란 그 함수가 든 프로그램이 실행 중인 런타임에 실행되는 함수가 아니라, 컴파일 중인 컴파일 타임에 실행되는 함수이다. 이러한 메타 함수를 작성하는 것이 템플리트 메타 프로그래밍이다. 학생들의 이해를 돕기 위해 다음의 URL들을 읽어보길 권한다.

   http://ko.wikipedia.org/wiki/템플릿_메타프로그래밍
   http://ricanet.com/new/view.php?id=blog/070303#none
   http://cdecl.tistory.com/6
   http://redo.egloos.com/249263
   http://redo.egloos.com/172276

   즉 하노이의 탑 알고리즘이 컴파일 타임에 이미 다 실행되고, 프로그램을 실행하면 정해진 값을 내놓게 프로그램을 작성하는 것이 본 문제가 요구하는 것이다. 그렇지 않고 프로그램 실행 시간에 하노이의 탑 알고리즘이 수행되는 경우, 점수를 주지 않는다.

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

... // 숙제는 이 부분을 채우는 것

int main()
{
  Hanoi<1,2,3,5>::move();
  return 0;
}
```

   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.

   (a) 한번에 하나의 원판만 옮길 수 있다.

   (b) 큰 원판이 작은 원판위에 있어서는 안된다.

2. (a) 다음과 같이 함수 템플리트를 이용하여 특정 데이터 형을 가지는 값들로 구성된 배열 내부의 자료들을 정렬하는 선택 정렬 (selection sort) 프로그램을 작성하라. 단순히 함수 템플리트만을 만드는 경우이므로, 메타 함수를 쓰지 않아야 한다.

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

template<typename T>
void selection_sort(T* array, int arr_size)
{
... // 숙제는 이 부분을 채우는 것
}

int main()
{
  int arr1[10] = { 10, 4, 11, 6, 7, 9, 1, 32, 9, 64 };
  for (int i=0;i<10;i++) cout << arr1[i] << " "; cout << endl;
  selection_sort(arr1, 10);
  for (int i=0;i<10;i++) cout << arr1[i] << " "; cout << endl;

  double arr2[20] =
  { 10., 14.3, 11.23, 6.65, 777., 90.1, 1.111, 32.5, 9.12, 64.,
    1.11, 2.33, 50., 34.11, 15.22, 3.4, 11.1, 218., 999., 21. };
  for (int i=0;i<20;i++) cout << arr2[i] << " "; cout << endl;
  selection_sort(arr2, 20);
  for (int i=0;i<20;i++) cout << arr2[i] << " "; cout << endl;

  char arr3[7] = { 'D', 'o', 'N', 'g', 'S', 'e', 'O' };
  for (int i=0;i<7;i++) cout << arr3[i] << " "; cout << endl;
  selection_sort(arr3, 7);
  for (int i=0;i<7;i++) cout << arr3[i] << " "; cout << endl;
  return 0;
}
```

출력 예는 다음과 같다.

```
10 4 11 6 7 9 1 32 9 64
1 4 6 7 9 9 10 11 32 64
10 14.3 11.23 6.65 777 90.1 1.111 32.5 9.12 64 1.11 2.33 50 34.11 15.22 3.4 11.1 218 999 21
1.11 1.111 2.33 3.4 6.65 9.12 10 11.1 11.23 14.3 15.22 21 32.5 34.11 50 64 90.1 218 777 999
D o N g S e O
D N O S e g o
```

(b) 템플리트 메타 프로그래밍을 이용하여 10 개의 정수 배열 내부의 자료들을 정렬하는 선택 정렬 (selection sort) 메타 함수를 작성하라.

```
#include <iostream>

using namespace std;

... // 숙제는 이 부분을 채우는 것

int main()
{
  int arr1[10] = { 10, 4, 11, 6, 7, 9, 1, 32, 9, 64 };
  for (int i=0;i<10;i++) cout << arr1[i] << " "; cout << endl;
  SelectionSort<10>::sort(arr1);
  for (int i=0;i<10;i++) cout << arr1[i] << " "; cout << endl;
}
```

출력 예는 다음과 같다.

```
10 4 11 6 7 9 1 32 9 64
1 4 6 7 9 9 10 11 32 64
```

힌트로 아래 URL의 버블 소트 알고리즘을 참고하라.

http://ubiety.uwaterloo.ca/~tveldhui/papers/Template-Metaprograms/meta-art.html

다시 말하지만, 본 문제는 버블 소트 알고리즘을 구현하라는 말이 아니라, 선택 정렬 알고리즘을 템플리트 메타 프로그래밍으로 구현하라는 뜻이다. 즉 컴파일 타임에 실행되는 메타 함수로 선택 정렬을 컴파일 타임에 다 실행해서 끝내 버리고, 실행 시간에는 결과만 출력하게 만들라는 것이다. 본 문제는 그렇게 프로그램을 작성하지 않으면 점수를 받을 수 없다.

(c) 앞의 두 문제에서 논의된 함수 템플리트에서의 데이터 형 매개 변수와 템플리트 메타 프로그래밍을 이용하여 특정 데이터 형을 가지는 값들로 구성된 배열 내부의 자료들을 정렬하는 선택 정렬 (selection sort) 프로그램을 작성하라.

```
#include <iostream>

using namespace std;

... // 숙제는 이 부분을 채우는 것

int main()
{
  int arr1[10] = { 10, 4, 11, 6, 7, 9, 1, 32, 9, 64 };
  for (int i=0;i<10;i++) cout << arr1[i] << " "; cout << endl;
  SelectionSort<int, 10>::sort(arr1);
  for (int i=0;i<10;i++) cout << arr1[i] << " "; cout << endl;

  double arr2[20] =
  { 10., 14.3, 11.23, 6.65, 777., 90.1, 1.111, 32.5, 9.12, 64.,
    1.11, 2.33, 50., 34.11, 15.22, 3.4, 11.1, 218., 999., 21. };
  for (int i=0;i<20;i++) cout << arr2[i] << " "; cout << endl;
  SelectionSort<double, 20>::sort(arr2);
  for (int i=0;i<20;i++) cout << arr2[i] << " "; cout << endl;

  char arr3[7] = { 'D', 'o', 'N', 'g', 'S', 'e', 'O' };
  for (int i=0;i<7;i++) cout << arr3[i] << " "; cout << endl;
  SelectionSort<char, 7>::sort(arr3);
  for (int i=0;i<7;i++) cout << arr3[i] << " "; cout << endl;

  return 0;
}
```

출력 예는 다음과 같다.

```
10 4 11 6 7 9 1 32 9 64
1 4 6 7 9 9 10 11 32 64
10 14.3 11.23 6.65 777 90.1 1.111 32.5 9.12 64 1.11 2.33 50 34.11 15.22 3.4 11.1 218 999 21
1.11 1.111 2.33 3.4 6.65 9.12 10 11.1 11.23 14.3 15.22 21 32.5 34.11 50 64 90.1 218 777 999
D o N g S e O
D N O S e g o
```

본 문제도 템플리트 메타 프로그래밍으로 구현하라는 의미이므로, 컴파일 타임에 실행되는 메타 함수로 실제 선택 정렬을 컴파일 타임에 다 실행해서 끝내 버리고, 실행 시간에는 결과만 출력하게 만들라는 것이다. 그렇게 프로그램을 작성하지 않으면 점수를 받을 수 없다.

3. (a) 다음과 같이 템플리트 메타 프로그래밍을 이용하여 십진수를 받아서 이진수로 출력하는 dec2bin 메타함수를 작성하라.

```
#include <iostream>

... // 숙제는 이 부분을 채우는 것

int main()
{
  std::cout << 0 << " : " << dec2bin<0>() << "\n";
  std::cout << 1 << " : " << dec2bin<1>() << "\n";
  std::cout << 2111165535 << " : " << dec2bin<2111165535>() << "\n";
  std::cout << 2147483647 << " : " << dec2bin<2147483647>() << "\n";
  std::cout << 2147483646 << " : " << dec2bin<2147483646>() << "\n";
  return 0;
}
```

출력 예는 다음과 같다.

```
0 : 0
1 : 1
2111165535 : 1111101110101011101010001011111
2147483647 : 1111111111111111111111111111111
2147483646 : 1111111111111111111111111111110
```

(b) 다음과 같이 템플리트 메타 프로그래밍을 이용하여 이진수를 받아서 십진수로 출력하는 bin2dec 메타함수를 작성하라.

```
#include <iostream>

... // 숙제는 이 부분을 채우는 것

int main()
{
  std::cout << "0" << " : " << bin2dec<0>::value << "\n";
  std::cout << "1" << " : " << bin2dec<1>::value << "\n";
  std::cout << "111" << " : " << bin2dec<111>::value << "\n";
  std::cout << "110011" << " : " << bin2dec<110011>::value << "\n";
  std::cout << "1100111" << " : " << bin2dec<1100111>::value << "\n";
  return 0;
}
```

출력 예는 다음과 같다.

```
0 : 0
1 : 1
111 : 7
110011 : 51
1100111 : 103
```

4. 다음과 같이 템플리트 메타 프로그래밍을 이용하여 주어진 데이터 형의 이름을 출력하는 type_printer 메타함수를 작성하라.

```
#include <iostream>

... // 숙제는 이 부분을 채우는 것

int main()
{
  std::cout << type_printer<int>() << "\n"
    << type_printer<char *>() << "\n"
    << type_printer<long const * &>() << "\n"
    << type_printer<short[10]>() << "\n"
    << type_printer<volatile int>() << "\n";
  return 0;
}
```

출력 예는 다음과 같다.

```
int
char *
long const * &
short [10]
int volatile
```

5. (a) cout 을 printf()로 구현하라. 이를 위해, my_ostream 이라는 클래스를 설계하고 my_cout 이라는 객체를 만들어 사용한다. my_ostream 클래스 내부에서 printf()를 사용하며, 물론 #include <iostream>과 그 헤더에 정의된 기능들을 사용하면 안된다.

   (b) printf()를 cout으로 구현하라. 이를 위해 가변적인 인수를 처리할 수 있는 my_printf() 함수를 cout을 사용하여 구현하되, 물론 #include <cstdio>, 또는 #include <stdio.h>와 그 헤더에 정의된 기능들을 사용하면 안된다.

6. 다음과 같이 출력 스트림에 대한 이터레이터 oIter 를 구현하라. 물론, #include <iterator>와 그 헤더에 정의된 기능들을 사용하면 안된다.

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

... // 숙제는 이 부분을 채우는 것

int main()
{
  oIter<char> outChar(cout);

  *outChar = 'X'; // 출력
  outChar++;
  *outChar = 'Y'; // 출력
  outChar++;
  *outChar = ' '; // 출력

  char str[] = "Dongseo University.";
  char *p = str;
  while(*p) *outChar++ = *p++;
  cout << endl;

  oIter<double> outDouble(cout);
  *outDouble = 187.23; // 출력
  outDouble++;
  *outDouble = -102.7; // 출력
  cout << endl;

  return 0;
}
```

출력 예는 다음과 같다.

```
XY Dongseo University.
187.23-102.7
```

7. dynamic_cast와 동일하게 동작하는 ptr_cast 템플리트를 구현하라. 다만, 이 ptr_cast 는 dynamic_cast 가 널 포인터(즉, 0)을 반환하는 경우에, 대신 bad_cast 예외를 날린다.

8. 주어진 정수 x가 있을 때, 0에서 x 까지의 정수를 쭈욱 출력할 때 나오는 '1'의 개수를 $f(x)$라고 하자. 예를 들어 $f(13)$을 구하기 위해 0부터 13까지 출력해 보면 다음과 같다.

   ```
   0 1 2 3 4 5 6 7 8 9 10 11 12 13
   ```

   위에서 1의 개수가 6 개 있으므로 $f(13) = 6$ 이다.

   $f(1) = 1$로 입력과 출력이 같다. 그렇다면 $f(n) = n$ 을 만족하는 모든 정수 n 들 중 1 보다 크면서 가장 작은 n 은 무엇인지 구하는 프로그램을 작성하라. (객체지향적으로 작성하지 않아도 됨)

9. http://en.wikipedia.org/wiki/Mandelbrot_set에 가보면 만델브로 집합을 복소수 (complex number) 좌표계에 그리기 위한 탈출 시간 알고리즘(escape time algorithm)이 나온다. 그 알고리즘의 대략적인 코드를 인용하면 다음과 같다.

```
For each pixel on the screen do:
{
  x = x0 = x co-ordinate of pixel
  y = y0 = y co-ordinate of pixel

  iteration = 0
  max_iteration = 1000

  while ( x*x + y*y <= (2*2)  AND  iteration < max_iteration )
  {

    xtemp = x*x - y*y + x0
    y = 2*x*y + y0

    x = xtemp

    iteration = iteration + 1
  }

  if ( iteration == max_iteration )
  then
    colour = black
  else
    colour = iteration

  plot(x0,y0,colour)
}
```

참고로, 만델브로 집합을 그릴 때, 복소수 좌표계에서의 최초의 경계는 보통 왼쪽 맨 위의 점은 $(-2, i)$, 오른쪽 맨아래의 점은 $(1, -i)$로 정한다.

(a) 만델브로 집합을 텍스트로 출력하는 프로그램을 작성하라. 출력 예는 코스 홈페이지를 참고하라.

(b) 만델브로 집합을 화면에 출력하는 프로그램을 작성하라. 출력 예는 코스 홈페이지를 참고하라.

10. Monte Carlo 방법으로 파이(π)를 구하는 프로그램을 작성하라. 이를 좀 더 설명하자면 다음과 같다.

    2차원 좌표계에 원점이 중심이고 반지름의 길이가 1인 원과, 그 원에 외접하는 한 변의 길이가 2이고 네 개의 변들이 X,Y 좌표 축에 평행한 정사각형을 생각해 보자. 그 정사각형은 $(-1,1)$, $(1,1)$, $(1,-1)$, $(-1,-1)$ 을 꼭지점으로 가지게 된다. 이제 -1에서 1 사이의 실수 x와 y를 랜덤으로 충분히 많이 생성한다. (예를 들어 10000 번 생성한다.) 생성한 점 (x,y)가 원 내부에 있으면 `count`를 증가시킨다. 최종적으로 `count` 값을 가지고 π를 구한다.

11. 많은 사람들이 인생은 줄을 잘 서야 한다고 한다. 이제 줄을 잘 서는 프로그램을 생각해 보자.

    m 명의 포로를 일렬로 세워서, k 번째 포로를 죽이고 그 자리에서 다음 k 번째 사람을 죽인다. 쭈욱 세어 나가다가 마지막 사람을 지나치면 처음 사람으로 간다. 그런 식으로 k 번째 사람을 계속 죽이다가, 맨 마지막에 남는 한 명을 살린다고 하자. 이 때 처음에 몇 번째 자리에 서야 살아남을 수 있는지 계산하는 프로그램을 짜 보자.

    이해를 돕기 위해, 예를 들면 다음과 같다.

    (a) m이 41이고 k가 3인 경우, 41 명의 포로가 있고 세번째 사람을 죽여 나가는 식이 된다. 이 경우 31번째 사람이 살아남는다.

    (b) m이 1이면 k에 상관 없이, 한 사람 밖에 없으므로 첫번째 사람이 살아남는다.

    (c) m이 2이고 k가 2인 경우, 첫번째 사람이 살아남는다.

    (d) m이 3이고 k가 2인 경우, 세번째 사람이 살아남는다.

    (e) m이 4이고 k가 2인 경우, 첫번째 사람이 살아남는다.

    (f) m이 5이고 k가 2인 경우, 세번째 사람이 살아남는다.

    (g) m이 6이고 k가 2인 경우, 5번째 사람이 살아남는다.

    (h) m이 7이고 k가 2인 경우, 7번째 사람이 살아남는다.

    (i) m이 8이고 k가 2인 경우, 첫번째 사람이 살아남는다.

    (j) m이 9이고 k가 2인 경우, 세번째 사람이 살아남는다.

    (k) m이 10이고 k가 2인 경우, 5번째 사람이 살아남는다.

    (l) m이 11이고 k가 2인 경우, 7번째 사람이 살아남는다.

    (m) m이 12이고 k가 2인 경우, 9번째 사람이 살아남는다.

    (n) m이 13이고 k가 2인 경우, 11번째 사람이 살아남는다.

    (o) m이 14이고 k가 2인 경우, 13번째 사람이 살아남는다.

    (p) m이 15이고 k가 2인 경우, 15번째 사람이 살아남는다.

    (q) m이 16이고 k가 2인 경우, 첫번째 사람이 살아남는다.

    프로그램은 m과 k라는 두 개의 정수를 입력 받는다. 물론 이 때, m은 k보다 크다. 그리고 프로그램은 죽기로 선택된 사람들을 차례로 출력한다. 실행 예는 다음과 같다.

```
m : 41  <-- 41 입력
k : 3   <-- 3 입력
3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 1, 5, 10, 14, 19, 23, 28, 32,
37, 41, 7, 13, 20, 26, 34, 40, 8, 17, 29, 38, 11, 25, 2, 22, 4, 35, 16, 31.
```

12. Suffix tree는 스트링을 색인하여 저장하기 위한 트리 데이터 구조이다. 이를 위해서 주어진 스트링의 마지막에 스트링의 끝을 나타내는 문자(주로 '$')를 붙인 후, 색인한다. 예를 들어 "banana"라는 스트링에 대한 suffix tree는 다음과 같다.

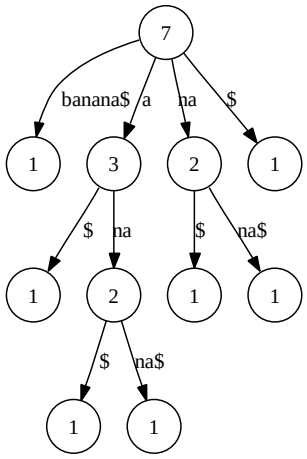

그림 1: "banana$" 스트링에 대한 suffix tree. 노드의 숫자는 패턴의 발생 빈도를 나타냄.

우선 주어진 "banana" 스트링 맨 뒤에 '$'를 붙인다. 그리고 나서, 접미어, 즉 suffix 들을 만들어서 하나씩 트리에 넣는다. 이러한 suffix 들은 "banana$", "anana$", "nana$", "ana$", "na$", "a$", 그리고 "$" 이다.

트리의 root 노드로부터 각각의 leaf 노드까지 가는 path는 하나의 suffix를 나타낸다. 특정 패턴이 주어지면, root 노드로부터 그 패턴을 따라간 path에서 가장 가까운 아래쪽으로 노드의 숫자가 그 패턴이 발생한 숫자이다. 예를 들면, "banana$"에서 "na" 패턴은 2 번 나타난다.

프로그램은 긴 스트링을 입력 받아서 suffix tree를 만들고, 패턴을 입력받아 그 패턴이 주어진 스트링에서 발생하는 개수를 출력한다. 만일 패턴이 발생하지 않는다면 물론 0을 출력한다.

Suffix tree는 리니어 타임(linear time)에 만들 수 있는 알고리즘이 있지만 본 과제에서는 그 정도를 원하지는 않고, 그냥 주어진 스트링에서 접미어들을 만들어서 트리에 그냥 삽입하여 구성하는 방법으로 평이하게 만들어도 된다.

PS: 너무 당연한 얘기지만, 만일 인터넷에서 떠도는 소스 코드를 전혀 이해하지도 못하면서 자기가 한 것인양 복사해서 보낸 사실이 드러날 경우, 본 과제 전체를 0 점 처리한다. 다시 말하지만, 인터넷에 떠도는 소스는 자신의 공부를 위해 참고만 가능하지 그 소스의 어떠한 부분도 자신이 제출할 프로그램에 넣어서는 안된다.

예를 들어 인터넷에서 떠도는 소스를 이해도 못하면서 다운 받아 저작권이나 만든 사람 이름 같은 것들만 지우고 제출한다거나, "XXX를 참고했

습니다"라고 쓰기만 하고 아무 생각 없이 똑같은 내용을 그대로 복사해서 제출하면 과제가 0점 처리될 뿐더러 죄질에 따라 본 과목을 F 처리할 것이다.

13. 가장 흔한 파일 압축 방법은 run length 방법이다. 그러나, 이 방법은 특정 문자나 패턴이 몇번 나타나는지를 저장하는 방법으로, 중고등학생도 알 정도로 평이하다.

    본 과제에선 허프만 코딩을 구현하는 파일 압축 및 해제 프로그램을 작성해 보자. 이를 위해서는 주어진 파일에서 문자들의 분포를 구해서 허프만 트리를 생성하여 이를 통해 가변 길이 코드 테이블을 만든다. 구성된 허프만 트리에 의한 가변 길이 코드 테이블을 통해서 주어진 파일을 가변 길이 코드로 인코딩한다. 그리고 나서 테이블과 새롭게 인코딩된 파일을 같이 새로운 압축 파일로 저장한다.

    만들어진 프로그램은 주어진 파일을 가지고 허프만 트리를 구성하여 가변 길이 코드 테이블로 압축을 하고 동시에 압축률을 출력한다. 예를 들어 100 바이트의 파일이 60 바이트가 되면 압축률은 60%이다. 주어진 파일이 크면 클수록, 그리고 텍스트 파일인 경우, 더 좋은 압축률을 보일 것이다.

    허프만 트리를 작성하는 방법은 다음의 URL을 참고하기 바란다.

    `http://kaistizen.net/project/Zip/Huffman.html`
    `http://myhome.naver.com/gginnie1/datacomm/huffman_coding.htm`

    PS: 바로 앞의 문제처럼 역시 만일 인터넷에서 떠도는 소스 코드를 대충 복사해서 보낸 사실이 드러날 경우, 과제 전체를 0 점 처리하거나 죄질에 큰 경우 본 과목에 대해 F 처리한다.