객체지향프로그래밍 숙제 #3 해답

강대기

2007년 12월 7일

제 1 절 숙제

1.1 연습 문제 숙제 (문제 2.3)

1. 연습문제 5.1, 5.9, 6.1, 6.2, 6.3, 6.9 는 교재의 부록 J를 참조하기 바란다.

2. Duff device는 Tom Duff가 고안해 낸 loop unrolling 방법이다. Loop unrolling이란 말 그대로 루프를 푸는 것인데, 일반적으로 루프로 구현되는 반복적인 동작을 루프를 없애고 단순히 명령을 여러개 복사해서 풀어쓰는 식으로 구현된다. 임베디드 프로그래밍이나 그래픽스 프로그래밍에서 빠르게 값들을 복사해야 할 때 이용될 수 있다.

   예를 들어 다음과 같이 특정 포트를 통해 디바이스에 연결된 레지스터에 스트링이나 배열의 값들을 복사하는 예를 보자.

   ```
   #define HAL_IO_PORT *(volatile char*)0xFFFF8000

   for (i = 0; i < len; ++i) {
     HAL_IO_PORT = *pSource++;
   }
   ```

   여기서는 루프를 사용하여 계속 pSource를 증가시키고 있다. 즉 한번 값을 쓰고 나서 루프의 처음으로 점프하고, 루프 카운터를 증가하고, 증가한 값을 길이 값과 비교한다.

   매번 이렇게 하는 대신 다음과 같이 반복해서 쓸 수 있다.

   ```
   HAL_IO_PORT = *pSource++;
   HAL_IO_PORT = *pSource++;
   HAL_IO_PORT = *pSource++;
   HAL_IO_PORT = *pSource++;
   ...
   ```

   이 경우에는 위에서 언급한 점프, 증가, 비교 등의 일을 하지 않는다. 물론 이 방법은 상당히 무식해 보인다. 즉 프로그램 내에 위의 `HAL_IO_PORT = *pSource++;`가 `len` 만큼 있는 것이다.

따라서, 위의 코드는 몇가지 문제가 있음을 바로 알 수 있다. 첫째로, 언제나 정해진 개수만큼만 복사할 수 있다. 루프의 경우처럼 때에 따라 복사할 길이가 변하는 경우에도 유연하게 적응할 수 없다. 둘째로, 코드의 크기가 엄청 증가할 수도 있다.

따라서 프로그래머들은 다음과 같이 일정한 양만큼만 코드를 복사하고 루프를 유지하는 방법을 사용하게 되었다.

```
int n = len / 8;
for (i = 0; i < n; ++i) {
  HAL_IO_PORT = *pSource++;
  HAL_IO_PORT = *pSource++;
  HAL_IO_PORT = *pSource++;
  HAL_IO_PORT = *pSource++;
  HAL_IO_PORT = *pSource++;
  HAL_IO_PORT = *pSource++;
  HAL_IO_PORT = *pSource++;
  HAL_IO_PORT = *pSource++;
}
```

위의 경우는 8 개가 있으므로, 처음의 루프의 경우에 비해 8 배 정도 루프 오버헤드가 감소한 셈이다. 그런데, 문제는 `len`이 8로 나누어 떨어지지 않는 경우이다. 이 경우, 나머지가 있게 되므로 위의 코드 바로 다음에 다음과 같은 코드를 추가해서 후처리를 해주어야 한다.

```
int n = len % 8;
for (i = 0; i < n; ++i) {
  HAL_IO_PORT = *pSource++;
}
```

1983년 루카스 필름에서 일하던 Tom Duff 가 위의 후처리 문제를 우아하게 해결한 기가 막힌 코드를 작성한다.

```
int n = (len + 8 - 1) / 8;
switch (len % 8) {
  case 0: do { HAL_IO_PORT = *pSource++;
  case 7:      HAL_IO_PORT = *pSource++;
  case 6:      HAL_IO_PORT = *pSource++;
  case 5:      HAL_IO_PORT = *pSource++;
  case 4:      HAL_IO_PORT = *pSource++;
  case 3:      HAL_IO_PORT = *pSource++;
  case 2:      HAL_IO_PORT = *pSource++;
  case 1:      HAL_IO_PORT = *pSource++;
             } while (--n > 0);
}
```

이제 후처리되는 나머지는 처음에 처리되게 된다. 처음에 `len`을 8로 나눈 나머지에 해당하는 `case`부터 코드는 시작하고, 마지막의 `while` 루프를 통해 나머지를 제외하고 8로 나누어 떨어지는 부분만큼이 수행된다.

3. 세미콜론을 안넣기 위해서는 다음과 같은 방법들이 가능하다.

   (a)
   ```
   #include <iostream>
   int main ()
   {
     if (std::cout << "Hello World\n") {}
   }
   ```

   (b)
   ```
   #include <iostream>
   int main ()
   {
     if (printf("Hello World\n")) {}
   }
   ```

1.1.1 중복되지 않는 난수들

단순히 난수를 생성하는 게 아니라, 기존의 수들을 무작위로 섞어서 2761 개의 수들을 제외하면 된다.

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

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

  int nums[32761] = {0};
  for (int i=0;i<32761;i++) nums[i]=i;

  for (int i=0;i<32761;i++)
  {
    int x=rand()%32761;
    int y=rand()%32761;
    int temp = nums[x];
    nums[x] = nums[y];
    nums[y] = temp;
  }
  for (int i=0;i<32760;i++) cout << nums[i] << " ";
  cout << endl << "The end!\n";
  return 0;
}
```

1.1.2 중첩된 /* */ 커멘트

다음과 같은 방법들이 가능하다.

1. ```
   #include <iostream>
   int main()
   {
     int x = 2 /*/*/* 0 */*/*/ - 1;
     if (1==x) std::cout << "중첩된 커맨트가 허용됩니다." << std::endl;
     else std::cout << "중첩된 커맨트가 허용되지 않습니다." << std::endl;
   }
   ```

2. ```
   #include <stdio.h>
   int main (void)
   {
     /*
       /*
        */
     puts ("Nested comments not supported.");
     goto end;
   //   *
   //   */
     puts ("Nested comments supported.");
     goto end;
   end:
     return 0;
   }
   ```

3. ```
   #include <iostream>
   void main()
   {
   /*/* */ std::cout << "중첩된 커맨드가 허용되지 않습니다.\n";
   // */ std::cout << "중첩된 커맨드가 허용됩니다.\n";
   }
   ```

4. ```
   #include <iostream>
   void main()
   {
     std::cout<<"중첩된 커맨트가 허용"/*/**/"되지 않습니다.\n";// */"됩니다.\n";
   }
   ```

1.2 프로그래밍 문제 숙제 (소스 파일들을 하나의 ZIP으로 제출)

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

위의 문제들에 대해서는 인터넷에서 다운받을 수 있는 답안들을 참고하기 바란다.

나머지 프로그램 문제들에 대해서는 소스 코드를 첨부하였다.