2011-03-14 作業解答

1. 在 C++的迴圈（for, while, do while）中要如何直接跳下一輪或是結束迴圈?

Within a C++ loop, “continue” imperative is used to skip to the next round while “break” imperative is used to terminate the loop. For for-loop, the next round starts at the increment block and for while- or do-while-loop, the next round starts at the “while” condition.

2. 何謂遞迴(recursion)? 請並舉出一個程式範例。

Recursion is for the definition of a function or procedure to refer to itself either directly or indirectly. Following is an example in Python for evaluating an expression from a simple algebra system consists of only addition: “+”; multiplication: “*”; and parenthesis: “()”.

```
def fac(s):
	i = 0
	while i < len(s) and s[i] in '0123456789': i = i + 1
	if i > 0: return int(s[:i]), s[i:]
	assert[s[0] == '(']
	v, s = exp(s[1:])
	assert[s[0] == ')']
	return v, s[1:]
def trm(s):
	v, s = fac(s)
	while len(s) and s[0] == '*':
		vv, s = fac(s[1:])
		v = v * vv
	return v, s
def exp(s):
	v, s = trm(s)
	while len(s) and s[0] == '+':
		vv, s = trm(s[1:])
		v = v + vv
	return v, s
```

3. 程式寫作(用 C++ string class)：輸入一 string，輸出其中字元'a'出現的次數。［隨意題］輸出所出現各種字元的次數。

Here we read a string from the standard input and count the number of 'a's.
C++:

```
#include <iostream>
using namespace std;
int main()
{
	string s;
	cout << "input: ";
	cin >> s;
	int count = 0;
	for (int i = 0; i < s.size(); i ++) {
		if (s[i] == 'a') count ++;
	}
	cout << "the number of 'a's is " << count << '\n';
	return 0
}
```

A Python script does not need to be compiled. However, you need to add “#!/usr/bin/env python” to the first line of the file and use “chmod +x your_python_file” under bash to make

it an executable.
Python:

```
#!/usr/bin/env python
print 'input: ',
s = raw_input()
print "the number of 'a's is", len([c for c in s if c == 'a'])
```

Extra: To count all characters, we setup an array of size 256 corresponding to all possible characters represented by a “char”. In C++, a “char” can be viewed as an integer (signed by default) with the range –128~127. We need to convert it to “unsigned char” (ranged 0~255) so that it can be used as an index of an array. (Though, regular ASCII strings consist of only positive “char”s.) Note that the values in an array are not initialize when the array is created in C++. Thus, we need to zero it out before using.
C++:

```
#include <iostream>
using namespace std;
int main()
{
	string s;
	cout << "input: ";
	cin >> s;
	int cnt[256];
	for (int i = 0; i < 256; i ++) cnt[i] = 0;
	for (int i = 0; i < s.size(); i ++) {
		int n = (unsigned char) s[i];
		cnt[n] ++;
	}
	for (int i = 0; i < 256; i ++) if (cnt[i]) {
		cout << "the number of '" << char(i)
			<< "'s is " << cnt[i] << '\n';
	}
	return 0;
}
```

Now, you can forget about the details when you have native “set” support.
Python:

```
#!/usr/bin/env python
print 'input: ',
s = raw_input()
for a in set(s):
	print "the number of '%c's is" % a, len([c for c in s if c == a])
```

4. 程式寫作：輸入一整數 n

- a. 輸出 Fibonacci 數列的前 n 項
- b. 輸出其 16 進位的表示字串
- c. 輸出以'#'字元繪製有 n 字元為底的等腰三角形

Fibonacci: This can be done with recursion but that usually won't be efficient due to duplicated calculations. The straighforward way is as follows.
C++:

```
#include <iostream>
using namespace std;
int main()
{
	int n;
	cout << "Input n: ";
	cin >> n;
	int p = 0;
	int q = 1;
```

```
        cout << "The Fibonacci Seq.:";
        for (int i = 0; i < n; i ++) {
              if (i) cout << ',';
              cout << ' ' << q;
              q += p;
              p = q - p;
        }
        cout << '\n';
        return 0;
  }
```

But, if you have to do it recursively, you can keep the last two terms in the sequence to avoid duplicated work.

C++:

```
  #include <iostream>
  using namespace std;
  void fib(int n, int & p, int & q)
  {
          if (n == 1) {
                  p = 0;
                  q = 1;
                  cout << ' ' << q;
                  return;
          }
          fib(n - 1, p, q);
          q += p;
          p = q - p;
          cout << ", " << q;
  }
  int main()
  {
          int n;
          int p, q;
          cout << "input n: ";
          cin >> n;
          cout << "The Fibonacci Seq.:";
          if (n > 0) fib(n, p, q);
          cout << '\n';
          return 0;
  }
```

We can pass around the whole sequence as a list in...

Python:

```
  #!/usr/bin/env python
  def fib(n):
          if n == 1:
                  return [0, 1]
          return (lambda a: a + [a[-1] + a[-2]])(fib(n - 1))
  print "input:",
  n=int(raw_input())
  print "The Fibonacci Seq.:", fib(n)[1:]
```

Hexadecimal: The conversion can be trivially done with the “hex” output manipulator: “cout << hex << n”. However, let's do it the hard way:

C++:

```
  #include <iostream>
  using namespace std;
  int main()
  {
        unsigned long n;
        cout << "Input n: ";
        cin >> n;
        char dgt[] = "0123456789ABCDEF";
```

```
        unsigned long m = 1;
        int e = 1;
        while (n / m >= 16) {
            m *= 16;
            e ++;
        }
        cout << n << " = 0x";
        while (e) {
            cout << dgt[(n / m) % 16];
            m /= 16;
            e --;
        }
        cout << '\n';
        return 0;
    }
```

Python:

```
    #!/usr/bin/env python
    print 'Input n:',
    n = int(raw_input())
    d = ''
    print n, '=',
    while n > 0:
        d = '0123456789ABCDEF'[n % 16] + d
        n /= 16
    print '0x' + d
```

Text triangle: This one takes a little counting...

C++:

```
    #include <iostream>
    using namespace std;
    int main()
    {
        int n;
        cout << "Input n: ";
        cin >> n;
        for (int i = 1; i < n + 1; i ++) {
            for (int j = i; j < n; j ++) cout << ' ';
            for (int j = 0; j < i; j ++) cout << " #";
            cout << '\n';
        }
        return 0;
    }
```

Python:

```
    #!/usr/bin/env python
    print 'Input n:',
    n = int(raw_input())
    for i in range(1, n + 1): print ' ' * (n - i) + ' #' * i
```