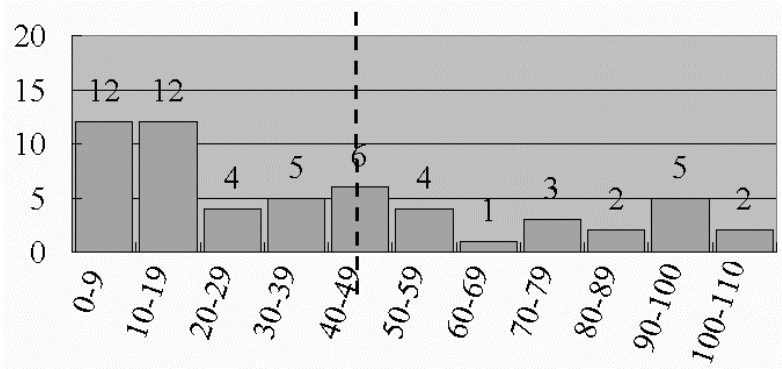
982 Midterm Exam Discussions

C++ Object Oriented Programming

Pei-yih Ting
99/05 NTOU CS

Score Statistics



Average: 43

1	1a	1b	1c	1d	1e	1f	2a	2b	2c	2d	3	4	5	6
53%	50%	57%	43%	41%	70%	31%	42%	27%	29%	38%	55%	36%	25%	35%

Problem #1

```
01 // RingBuf.h: interface for the CRingBuf class.
02
03 #ifndef RINGBUF_H
04 #define RINGBUF_H
05
06 class CRingBuf
07 {
08 public:
09     bool resize(const int newSize);
10     CRingBuf& operator=(const CRingBuf &rhs);
11     int dataCount() const;
12     bool get(int &data);
13     bool put(const int data);
14
15     int write(const int dataArray[], const int dataCount);
16     int read(int dataArray[], const int dataCount);
17
18     static void unitTest();
19
20     CRingBuf(const int bufSize);
21     CRingBuf(const CRingBuf &src);
22  ~CRingBuf();
23
24 private:
25     int m_bufSize;
26     int m_head, m_dataCount;
27     int *m_buf;
28 };
29
30 #endif
```


#1 (a) Constructor (b) Destructor

```
015 CRingBuf::CRingBuf(const int bufSize):
016     m_bufSize(bufSize),
017     m_head(0), m_dataCount(0),
018     m_buf(0)
019 {
020     if (m_bufSize > 0)
021         m_buf = new int[m_bufSize];
022     else
023         m_bufSize = 0;
024 }

058 CRingBuf::~CRingBuf()
059 {
060     delete[] m_buf;
061 }
```

#1 (c) get() (d) put()

```
075 bool CRingBuf::get(int &data)
076 {
077     if (m_dataCount > 0)
078     {
079         data = m_buf[m_head];
080         m_head = (m_head+1) % m_bufSize;
081         m_dataCount--;
082         return true;
083     }
084     else
085         return false;
086 }
```

```
063 bool CRingBuf::put(const int data)
064 {
065     if (m_dataCount < m_bufSize)
066     {
067         m_buf[(m_head+m_dataCount)%m_bufSize] = data;
068         m_dataCount++;
069         return true;
070     }
071     else
072         return false;
073 }
```


#1 (e) dataCount()

```
088 int CRingBuf::dataCount() const
089 {
090     return m_dataCount;
091 }
```


#1 (f) unitTest()

```
130 void CRingBuf::unitTest()
131 {
132     const int rbsize=30;
133     CRingBuf rbuf(rbsize), *tmpbuf;
134     int i, count=0, error, sum;
135     int temp;
136     int data[30] = {100, 101, 102, 103, 104, 105, 106, 107, 108, 109};
137
138     srand(0);
139     // basic put test and overflow test
140     for (i=0; i<rbsize; i++)
141         assert(rbuf.put(count++));
142     assert(rbuf.dataCount() == rbsize);
143     for (i=0; i<10; i++)
144         assert(!rbuf.put(count++));
145     assert(rbuf.dataCount() == rbsize);
146     cout << "basic put() test ok\n";
```


#1 (f) unitTest() (cont'd)

```
148     // copy ctor and basic get test
149     tmpbuf = new CRingBuf(rbuf);
150     for (i=0; i<rbsize; i++)
151     {
152         assert(tmpbuf->get(temp));
153         assert(temp == i);
154     }
155     assert(tmpbuf->dataCount() == 0);
156     delete tmpbuf;
157
158     for (i=0; i<rbsize; i++)
159     {
160         assert(rbuf.get(temp));
161         assert(temp == i);
162     }
163     assert(rbuf.dataCount() == 0);
164     cout << "copy ctor and basic get() test ok\n";
```

#1 (f) unitTest() (cont'd)

```
166   // around the buffer test and underflow test
167   error = 0;
168   for (i=0; i<rbsize*20; i++)
169   {
170      assert(rbuf.put(i));
171      assert(rbuf.get(temp));
172      assert(temp == i);
173      assert(!rbuf.get(temp));
174      error++;
175   }
176   assert(error == rbsize*20);
177   cout << "around the buffer test ok\n";
```


#1 (f) unitTest() (cont'd)

```
179   // random test
180   assert(rbuf.dataCount() == 0);
181   error = 0;
182   for (i=0; i<rbsize*20; i++)
183   {
184      if (rand()%5 == 0)
185         if (!rbuf.put(i)) error++;
186      if (rand()%6 == 0)
187         if (!rbuf.put(i*2)) error++;
188      if (rand()%7 == 0)
189         if (!rbuf.get(temp)) error++;
190   }
191 //    cout << error << ' ' << rbuf.dataCount() << endl;
192   assert(error == 82);
193   assert(rbuf.dataCount() == 28);
```


#1 (f) unitTest() (cont'd)

```
194
195   sum = 0;
196   count = rbuf.dataCount();
197   for (i=0; i<count; i++)
198   {
199      assert(rbuf.get(temp));
200      sum += temp;
201   }
202 //    cout << "sum = " << sum << endl;
203   assert(sum == 21361);
204   assert(rbuf.dataCount() == 0);
205   cout << "random put()/get() test ok\n";
```


Problem #2 Big 3

- The Big 3:
  - Destructor: ~CRingBuf()
  - Copy Constructor: CRingBuf(CRingBuf &)
  - Assignment Operator: CRingBuf& operator=(CRingBuf&)
- They appear togethor when your class allocates some resources (e.g. memory).

#2 (a) copy ctor (b) where is it used

(a) Copy constructor

```
026 CRingBuf::CRingBuf(const CRingBuf &src):
027     m_buf(0),
028     m_bufSize(src.m_bufSize),
029     m_dataCount(src.m_dataCount),
030     m_head(src.m_head)
031 {
032     int i;
033     m_buf = new int[m_bufSize];
034     for (i=0; i<m_bufSize; i++)
035         m_buf[i] = src.m_buf[i];
036 }
```

(b) Where is it used?

i. Initialize an object from another object CRingBuf x; CRingBuf y(x);

ii. Call-by-value CRingBuf someFunction(CRingBuf x) { … }

iii. Return-by-value

#2 (c) operator= (d) default?

(c) Assignment operator

```
038 CRingBuf& CRingBuf::operator=(const CRingBuf &rhs)
039 {
040     if (&rhs == this)
041         return *this;
042
043     delete[] m_buf;
044
045     m_bufSize = rhs.m_bufSize;
046     m_dataCount = rhs.m_dataCount;
047     m_head = rhs.m_head;
048
049     int i;
050     m_buf = new int[m_bufSize];
051     for (i=0; i<m_bufSize; i++)
052         m_buf[i] = rhs.m_buf[i];
053
054     return *this;
055 }
```

(d) Default behavior if not defined?

If you do not define operator=() in your class, C++ compiler synthesize a bit-wise copy assignment operator for your class. i.e. If you use CRingBuf x, y; ….. x = y; C++ compiler would do a bit-wise copy from object y to object x.

In a class like CRingBuf, this would cause object x and object y share the same memory. When one object is destructed earlier than the other, it cause a situation called “dangling reference” in which the pointed memory by the remaining object is already deleted.

Problem #3 write()

```
093 int CRingBuf::write(const int dataArray[], const int dataCount)
094 {
095     int i;
096     for (i=0; i<dataCount; i++)
097         if (!put(dataArray[i])) break;
098     return i;
099 }
100
101 int CRingBuf::read(int dataArray[], const int dataCount)
102 {
103     int i;
104     for (i=0; i<dataCount; i++)
105         if (!get(dataArray[i])) break;
106     return i;
107 }
```


Problem #4 resize()

```
109 bool CRingBuf::resize(const int newSize)
110 {
111     int i;
112     if (m_dataCount > newSize)
113         return false;
114     else if (m_dataCount == newSize)
115         return true;
116
117     int *buf = new int[newSize];
118
119     for (i=0; i<m_dataCount; i++)
120         buf[i] = m_buf[(m_head+i)%m_bufSize];
121
122     delete[] m_buf;
123     m_buf = buf;
124     m_bufSize = newSize;
125     m_head = 0;
126
127     return true;
128 }
```

Problem #5 array of objects

- Require a “default constructor”

```
CRingBuf::CRingBuf(): m_bufSize(0), m_head(0), m_dataCount(0), m_buf(0)
{
}
```

- Or use the default argument of the constructor function

```
class CRingBuf {
…
public:
   CRingBuf(const int bufSize = 0);
…
};
```


Bonus Problem: more unitTest()

```
207  // write() test
208  i = rbuf.write(data, 10);
209  assert((i == 10) && (rbuf.dataCount() == 10));
210  for (i=0; i<10; i++)
211  {
212     assert(rbuf.get(temp));
213     assert(temp == data[i]);
214  }
215
216  i = rbuf.write(data, 10);
217  assert((i == 10) && (rbuf.dataCount() == 10));
218  i = rbuf.write(data, rbsize);
219  assert((i == rbsize-10) && (rbuf.dataCount() == rbsize));
220  cout << "write() interface test ok\n";
```


more unitTest() (cont’d)

```
230  // operator= test
231  i = rbuf.write(data, 17);
232  assert((i == 17) && (rbuf.dataCount() == 17));
233  CRingBuf *rbuf1 = new CRingBuf(5);
234  *rbuf1 = rbuf;
235  assert(rbuf.dataCount() == 17);
236  assert(rbuf1->dataCount() == 17);
237  for (i=0; i<17; i++)
238  {
239     assert(rbuf1->get(temp));
240     assert(temp == data[i]);
241     assert(rbuf.get(temp));
242     assert(temp == data[i]);
243  }
244  delete rbuf1;
245  cout << "operator=() interface test ok\n";
```


more unitTest() (cont’d)

```
247  // resize() test
248  i = rbuf.write(data, 20);
249  i = rbuf.read(data, 20);
250  i = rbuf.write(data, 15);
251  i = rbuf.read(data, 5);
252  assert((i == 5) && (rbuf.dataCount() == 10));
253  rbuf.resize(15);
254  for (i=0; i<10; i++)
255  {
256     rbuf.get(temp);
257     assert(temp == data[i+5]);
258  }
259
```

more unitTest() (cont'd)

```
260   i = rbuf.write(data, 20);
261   i = rbuf.read(data, 20);
262   i = rbuf.write(data, 15);
263   i = rbuf.read(data, 5);
264   assert((i == 5) && (rbuf.dataCount() == 10));
265   rbuf.resize(50);
266   for (i=0; i<10; i++)
267   {
268      rbuf.get(temp);
269      assert(temp == data[i+5]);
270   }
271   cout << "resize() interface test ok\n";
```


Summary

- 本次考試主要評估下列概念與語法：
  - 物件
  - 建構元/解構元/初始化串列的設計與呼叫時機
  - static member function (class function)
  - C/C++ 中字串與字元陣列的表示方法與使用方法
  - 封裝
  - 類別的設計
  - const 的使用 (參數, 成員函式)
  - 動態配置記憶體 new / delete
  - 基本迴圈練習
  - unitTest
  - assert

Problems Found

1. 還不能夠短時間寫一段 20-30 列可以正常運作的程式
2. C/C++ 資料型態系統不清楚
3. 看到 C/C++ 指標/參考就昏了
4. 現在連陣列都列入討厭的對象
5. C++ 類別/成員函式/建構元/解構元語法不熟悉
6. 不喜歡 assert
7. 自己寫的功能沒有思考怎樣測試
8. 沒有站在物件的角度去想怎樣寫物件的成員函式, 常常忘了成員函式可以直接存取物件私有的資料成員

每次在寫程式的時候, 不要照著參考範例抄, 至少應該先看過, 瞭解該語法的意義與模型以後, 把資料蓋起來, 盡量讓自己練習把運作的邏輯用所學到的語法轉換爲程式, 熟悉以後才能夠應用