2018-2019 OOP 期末考试

True-or-False 10 | A. Multiple-Choice - 1 10 | Fill-in-Blank 9 | Fill-in-Blank - P 3

5-1 The function template printArrayInfo() computes the minimal, maximal and average value of a two dimension array and prints them out, where nrows is number of rows and ncols is the number of columns.

Author: hulanqing
Organization: 浙江大学
Time Limit: 400 ms
Memory Limit: 64 MB

```
#include <iostream>

template<class T>          (1分)
void printArrayInfo(T*          (1分) array, int nrows, int ncols)
{
  T                        (1分) max = array[0], min = array[0];
  double avg = 0           (1分);
  for(int i = 0; i < nrows; ++i)
  {
      for(int j = 0; j < ncols; ++j)
      {
          T val            (1分) = array[i*ncols+j]          (1分);
          if(val<min       (1分))  min = val;
          if(val>max       (1分))  max = val;
          avg =avg+static_cast<double>(v  (1分);
      }
  }
  avg /= (nrows*ncols      (1分));
  std::cout << "min=" << min << std::endl;
  std::cout << "max=" << max << std::endl;
  std::cout << "avg=" << avg << std::endl;
}

int main()
{
  int ai[2][3]={{8,10,2},{14,4,6}};
  printArrayInfo(ai[0], 2, 3);
  double af[1][5]={{3.4f,4.2f,6.6f,2.4f,-0.9f}};
  printArrayInfo(af[0], 1, 5);
  return 0;
}
```

5-1 Accepted (10 point(s))

5-2 The class String is a simple C++ encapsulation of the C character arrays.

Author: hulanqing
Organization: 浙江大学
Time Limit: 400 ms
Memory Limit: 64 MB

```
#include <cstring>
#include <iostream>
#include <stdexcept>

class StringIndexError : public std::out_of_range {
private:
    int index;
public:
    StringIndexError(int idx) : std::out_of_range(""), index(idx) {}
    int getIndex() const
    {
       return index;
    }
};

class String {
private:
    char *m_ptr;
public:
    String(const char *ptr)
    {
        m_ptr = new char[strlen(ptr)+1]          (1分);
        strcpy(m_ptr, ptr);
    }
    ~String()
    {
        delete[] m_ptr                           (1分);
    }
    String &operator+=(const String &str)
    {
        char *s = new char[strlen(m_ptr)+strlen(str.m_ptr)+1]   (1分);
```

```
    if (m_ptr)
    {
        strcpy(s, m_ptr);
        delete[]                                    (1分) m_ptr;
    }
   strcat(s, str.m_ptr); // appends str.m_ptr to s
   m_ptr                                            (1分) = s;
   return *this                                     (1分);
}
bool operator==(const String &str) const
{
    return (strcmp(m_ptr, str.m_ptr) == 0);
}
char& operator[](int i)
{
    if (i >= 0 && i < strlen(m_ptr)) return m_ptr[i];
    throw StringIndexError(i);
}
   friend                                           (1分) std::ostream& operator<<(std::ostream &, const String &);
};

std::ostream&                                       (1分) operator<<(std::ostream &out, const String &str)
{
    return out << str.m_ptr;
}

int main()
{
    String s1("Hello "), s2("world!");

    if (s1 == s2)
        std::cout << "S1==S2" << std::endl;
    else
        std::cout << "S1!=S2" << std::endl;

    s1 += s2;
    std::cout << s1 << std::endl;

    try                                             (1分) {
        int k = 0;
        while (true)
          std::cout << s1[k++];
    }
    catch                                           (1分) (const StringIndexError& ex) {
        std::cout << "\nString index is out of range: " << ex.getIndex() << std::endl;
    }

    return 0;
}
```

5-2 Accepted (10 point(s))

5-3 The class Queue implements a circular queue data structure.

Author: hulanqing
Organization: 浙江大学
Time Limit: 400 ms
Memory Limit: 64 MB

```
#include <iostream>

template<class T>
class Queue {
private:
  int capacity;         // capacity of the queue
  T*                                (1分)data;              // dynamically allocated array of doubles
  int front;            // head of the queue
  int rear;             // tail of the queue
public:
  Queue(int maxsize);
  ~Queue();
  bool empty();
  bool full();
  void push(T a);       // append a double value to the tail of queue
  T pop();                      // delete the head element of the queue
};

template<class T> Queue<T>::Queue(int maxsize)
{
  capacity = maxsize;
  data = new T[maxsize]                             (1分);
```

```
  front = rear = 0;
  std::cout << "queue initialized! ";
}
template<class T> Queue<T>::~Queue()
{
  delete[] data                          (1分);
  std::cout << "queue destroyed! ";
}
template<class T> bool Queue<T>::empty()
{
  return (front == rear)                 (1分);
}
template<class T> bool Queue<T>::full()
{
  return (front == ((rear+1)%capacity))  (1分);
}

//The dynamic array data will be a circular Queue
template<class T> void Queue<T>::push(T a)
{
  if (full())
  {
        exit(0);
  }
  else
  {
    data[rear]                           (1分) = a;
    rear = (rear+1)%capacity             (1分);
  }
}

template<class T> T Queue<T>::pop()
{
  if (empty())
  {
        exit(0);
  }
  T top=data[front]                      (1分);
  front = (front+1)%capacity             (1分);
  return top;
}

int main()
{
  Queue<double>                          (1分) q(5);
  std::cout << q.empty();

  q.push(1.3);
  q.push(2.3);
  q.push(3.3);
  q.push(4.3);
  std::cout << q.full();

  q.pop();
  q.pop();
  q.pop();
  q.push(5.3);
  q.push(6.3);
  q.push(7.3);
  std::cout << q.full();

  q.pop();
  q.pop();
  q.pop();
  q.pop();
  std::cout << q.empty();
  return 0;
}
```

5-3 Accepted (10 point(s))