1.

Code:

#include <iostream>

using namespace std ;

template <typename T>

T max(T a, T b, T c)

{

return (a > b ? a : b) > c ? (a > b ? a : b) : c ;

}

void main()

{

int a = 41, b = 0 ;

double c = 4.1 ;

cout << max<double>(a, b, c) << endl ;

short a1 = 3 ;

char b1 = 2 ;

long c1 = 99 ;

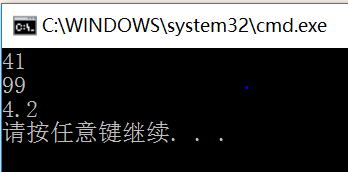
cout << max<long>(a1, b1, c1) << endl ;

float a2 = 4.2f ;

cout << max<double>(b, a2, c) << endl ;

}

Result:



2.

Code:

#include <iostream>

using namespace std ;

template <class T>

class Queue

{

private:

T\* Front, \* Tail ;

int max ;

public:

Queue(int m) : max(m)

{

Front = new T[m + 1] ;

Tail = Front ;

}

Queue(Queue &q) : max(q.max)

{

Front = new T[max + 1] ;

int i = 0 ;

for (; i < q.Tail - q.Front ; i++)

{

Front[i] = q.Front[i] ;

}

Tail = &Front[i] ;

}

~Queue()

{

delete []Front ;

}

void In(const T &a) ;

void Out(int num) ;

bool Empty() const ;

void Display() const ;

} ;

template <class T>

void Queue <T>::In(const T &a)

{

if(Tail == &Front[max])

{

cout << "Full! " << a << " input failed!" << endl ;

}

else

{

\*Tail = a ;

Tail += 1 ;

}

}

template <class T>

void Queue <T>::Out(int num)

{

if(Empty())

{

cout << "Empty!" << endl ;

}

else if(num > Tail - Front)

{

cout << "Out of range!" << endl ;

}

else

{

cout << Front[num - 1] << " out" << endl ;

for (int i = num - 1; i < Tail - Front - 1; i++)

{

Front[i] = Front[i + 1] ;

}

Tail-- ;

}

}

template <class T>

bool Queue <T>::Empty() const

{

return (Front == Tail + 1) ;

}

template <class T>

void Queue<T>::Display() const

{

T\* p = Front ;

if(Empty())

{

cout << "Empty!" << endl ;

}

else

{

for (; p < Tail; p++)

{

cout << \*p << ' ' ;

}

}

cout << endl ;

}

void main()

{

Queue <double> doubleq(10) ;

double j = 0 ;

for(int i = 0; i < 10; i++)

{

doubleq.In(j) ;

j += 1.0 ;

}

doubleq.Display() ;

doubleq.Out(2) ;

doubleq.Display() ;

Queue <int> intq(5) ;

intq.In(2) ;

intq.In(4) ;

intq.In(7) ;

intq.In(1) ;

intq.In(9) ;

intq.In(10) ;

Queue <int> intq2(intq) ;

intq2.Display() ;

intq2.Out(3) ;

intq2.Display() ;

}

Result:

0 1 2 3 4 5 6 7 8 9

1 out

0 2 3 4 5 6 7 8 9

Full! 10 input failed!

2 4 7 1 9

7 out

2 4 1 9

请按任意键继续. . .