

```

doc.cpp

#ifndef _Deque
#define _Deque
#include <vector>
using namespace std;
template <class T>
class Deque : public std::vector<T>
{
private:
public:
    Deque(void);
    Deque(const int _nSize, const int _nIncrement);
    virtual ~Deque(void);
    virtual int Push(const T &oValue);
    virtual int Pop(T &pValue);
    virtual int PushFirst(const T &oValue);
    virtual int PopBack(T &pValue);
};

template <class T>
Deque<T>::Deque(
    void
): std::vector<T>::vector()
{
    return ;
}

template <class T>
Deque<T>::Deque(
    const int _nSize
    , const int _nIncrement
): std::vector<T>::reserve(_nSize)
{
    return ;
}

template <class T>
Deque<T>::~Deque(
    void
)
{
    return ;
}

template <class T>
int Deque<T>::Push(
    const T &oValue
)
{
    int nStatus;
    nStatus = 0;
    nStatus = std::vector<T>::push_back(_oValue);
    return nStatus;
}

template <class T>
int Deque<T>::Pop(
    T &pValue
)
{
    int nStatus;
    nStatus = 0;
    _pValue=std::vector<T>::front();
    std::vector<T>::erase[std::vector<T>::begin()];
    return nStatus;
}

template <class T>
int Deque<T>::PushFirst(
    const T &oValue
)
{
    int nStatus;
    nStatus = 0;
    nStatus = std::vector<T>::insert(std::vector<T>::begin(), _oValue);
    return nStatus;
}

template <class T>
int Deque<T>::PopBack(
    T &pValue
)
{
    int nStatus;
    nStatus = 0;
    nStatus = std::vector<T>::pop_back(_pValue);
    return nStatus;
}

#endif

```