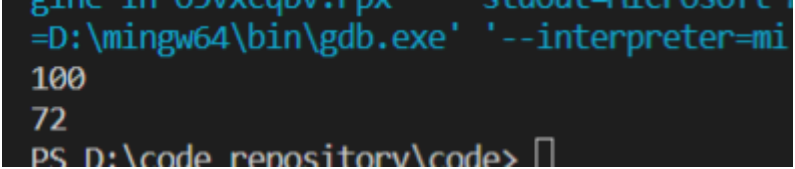


数据结构与算法 课程实验报告

学号：202200130048	姓名：陈静雯	班级：6
实验题目：队列		
实验学时：2	实验日期：11.9	
<p>实验目的：</p> <p>1、掌握队列结构的定义与实现；</p> <p>2、掌握队列结构的使用。</p>		
<p>软件开发工具：</p> <p>Vscode</p>		
<p>1. 实验内容</p> <p>1、题目描述：</p> <p>首先创建队列类，采用数组描述；实现卡片游戏，假设桌上有一叠扑克牌，依次编号为 1-n（从最上面开始）。当至少还有两张的时候，可以进行操作：把第一张牌扔掉，然后把新的第一张放到整叠牌的最后。输入 n，输出最后剩下的牌。</p> <p>2. 数据结构与算法描述（整体思路描述，所需要的数据结构与算法）</p> <p>先 pop 队首，再将 front 值 push 到队尾，最后剩一张牌，输出</p> <p>Queuefront 为队首的前一位，queueback 指向队尾元素</p> <p>Pop：循环队列，queuefront 的下一位为队首，将它置 0</p> <p>Push：queuefront==(queueback+1)%queue length 为队满，重新分配队列空间，将其扩充至两倍，再将元素插入队尾</p> <p>3. 测试结果（测试输入，测试输出）</p>  <p>4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）</p> <p>无问题</p> <p>5. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）</p> <pre>#include <iostream> using namespace std; template<class T> class myqueue{</pre>		

```

public:
    myqueue(int n=100);
    void init(int n);
    bool empty();
    void push(T& thelement);
    void pop();
    T front();
    T back();
    int size();
private:
    int queuefront;
    int queueback;
    int queuelength;
    int queuesize;
    T* element;
};

template<class T>
myqueue<T>::myqueue(int n) {
    element=new T [n];
    queuefront=n-1;
    queueback=n-1;
    queuelength=n;
    queuesize=0;
}

template<class T>
void myqueue<T>::init(int n) {    //按题目初始化队列
    for(int i=1;i<=n;i++) {
        element[i-1]=i;
    }
    queuefront=queuelength-1;
    queueback=n-1;
    queuesize=n;
}

template <class T>
bool myqueue<T>::empty() {
    return queuefront == queueback;
}

template <class T>
void myqueue<T>::push(T& thelement) {
    if(queuefront==(queueback+1)%queuelength) {
        //若空间不够,
        重新进行动态分配
        queuelength*=2;
    }
}

```

```

        T* temp = new T [queue length];
        for (int i=0; i<queue length; i++) {
            temp[i]=element[i];
        }
        element=temp;
    }
    queueback = (queueback+1)%queue length;
    element[queueback]=thelement;
    queuesize++;
}

template<class T>
void myqueue<T>::pop() {
    queuefront=(queuefront+1)%queue length;
    element[queuefront]=0;
    queuesize--;
}

template<class T>
T myqueue<T>::front() {
    return element[(queuefront+1)%queue length];
}

template<class T>
T myqueue<T>::back() {
    return element[queueback];
}

template<class T>
int myqueue<T>::size() {
    return queuesize;
}

int main() {
    int n;
    cin>>n;
    myqueue<int> a(n+5);
    a.init(n);
    while(a.size()>1) {
        a.pop();
        int i=a.front();
        a.pop();
        a.push(i);
    }
    int ans=a.front();
    cout<<ans;

```

}