Лабораторная работа №3 (3-й семестр)

> *Как видно, совершенство достигается не тогда, когда уже нечего прибавить, но когда уже ничего нельзя отнять.*
>
> — Antoine de Saint-Exupéry

Теоретические сведения

О́чередь — структура данных с типом доступа к элементам «первый пришёл — первый вышел» (FIFO, First In — First Out). Добавление элемента возможно лишь в конец очереди, а извлечение — только из начало очереди, при этом выбранный элемент из очереди удаляется. В различных библиотеках методы добавления и извлечения элементов в очередь могут называться по-разному. Часто для добавления используют название методов push или enqueue, а для извлечения — pop или dequeue (рис. 1).

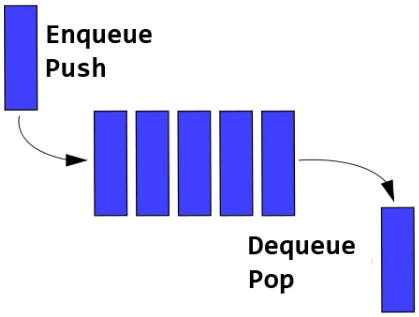

Рис. 1 — Графическое представление очереди

Пример. Написать программу, реализующею очередь для хранения целых чисел. Пользователь может помещать и извлекать данные из очереди. Содержимое очереди связывается с некоторым компонентом таким образом, чтобы пользователь мог видеть ее актуальное состояние.

Файл: *QueueMain.cpp*

```
...
#include "Queue.h"
...
Queue *q;
QueueFrame::QueueFrame(wxWindow* parent,wxWindowID id)
{
    ...
    q = new Queue(10);
    q->setWxGrid(Grid1);
}
QueueFrame::~QueueFrame()
{
    delete q;
}
void QueueFrame::OnPushBtnClick(wxCommandEvent& event)
{
    int elem = wxAtoi(TextCtrl1->GetValue());
    if (!q->push(elem))
        wxMessageBox(_("Переполнение"));
}
void QueueFrame::OnPopBtnClick(wxCommandEvent& event)
{
    int elem;
```

```
    if (q->pop(elem))
        TextCtrl1->SetValue(wxString()<<elem);
    else
        wxMessageBox(_("Стек пуст"));
}
```

Файл: *Queue.h*

```
#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED
#include <wx/grid.h>
class Queue
{
     int *q;              // Очередь
     int qEnd;            // Текущий размер очереди
     int maxQLength;      // Максимальный размер очереди
     wxGrid *wxg;         // wxGrid для отображения
public:
     Queue(int MaxLength);// Конструктор
     ~Queue();            // Деструктор
     bool push(int elem); // Добавление элемента в очередь
     bool pop(int & elem);// Извлечение элемента из очереди
     bool isEmpty();      // Очередь пуста?
     bool isFull();       // Очередь заполнена?
     void setWxGrid(wxGrid* _wxg);// Вывод очереди
};
#endif // QUEUE_H_INCLUDED
```

Файл: *Queue.cpp*

```
#include "Queue.h"
Queue::Queue(int MaxLength)
{
     qEnd = 0;
     maxQLength = MaxLength;
     q = new int[maxQLength];
     wxg = NULL;
}
Queue::~Queue()
{
     delete []q;
}
bool Queue::isEmpty()
{
     return qEnd == 0;
}
bool Queue::isFull()
{
     return qEnd == maxQLength;
}
bool Queue::push(int elem)
{
     if(isFull())
          return false;
     q[qEnd] = elem;     // Записываем новый элемент
     qEnd++;
```

```
     if (wxg)
     {
          wxg->InsertRows();
          wxg->SetCellValue(0,0,wxString()<<elem);
     }
     return true;
}
bool Queue::pop(int &elem)
{
     if(isEmpty())
          return false;
     elem = q[0];   // Записываем в elem извлекаемый э-т
     for(int i = 0; i < qEnd - 1; i++) // Сдвиг оставшихся э-тов
          q[i] = q[i + 1];
     qEnd--;   // Уменьшаем длину очереди
    if (wxg)
        wxg->DeleteRows(qEnd);
     return true;
}
void Queue::setWxGrid(wxGrid* _wxg)
{
     wxg = _wxg;
}
```

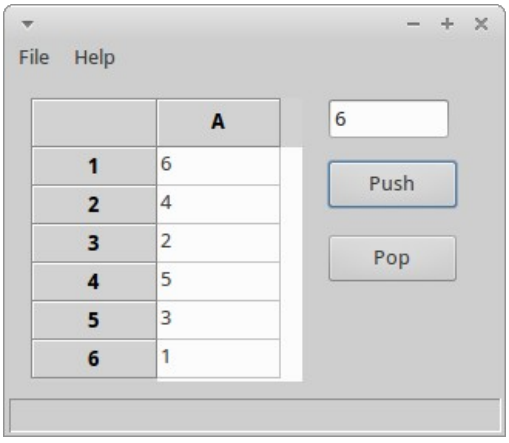

Рис. 2 — Пример работы приложения