Лабораторная работа №2
(3-й семестр)

Если вы лжете компилятору, то он будет мстить.

— Henry Spencer

Теоретические сведения

Стек (англ. stack — стопка) — структура данных, в которой доступ к элементам организован по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

Добавление элемента, называемое также проталкиванием (push), возможно только в вершину стека (добавленный элемент становится первым сверху). Удаление элемента, называемое также выталкиванием (pop), тоже возможно только из вершины стека, при этом второй сверху элемент становится верхним (рис. 1).

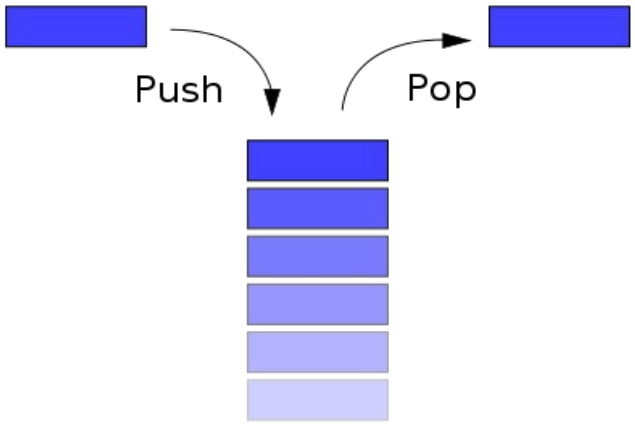

Рис. 1 — Графическое представление стека. Изображение взято с http://ru.wikipedia.org/wiki/Стек

Стеки широко применяются в вычислительной технике. Например, для отслеживания точек возврата из подпрограмм используется стек вызовов, который является неотъемлемой частью архитектуры большинства современных процессоров. Языки программирования высокого уровня также используют стек вызовов для передачи параметров при вызове процедур.

Для визуализация хода процесса выполнения в wxWidgets часто используется компонент **wxGauge**. Отображение хода процесса можно осуществлять, задавая значение позиции с помощью функции SetValue, а диапазон возможный значений с помощью функции SetRange. Например, если полная длительность процесса характеризуется значением целой переменной Count (объем всех копируемых файлов, число настроек, количество циклов какого-то процесса), а выполненная часть — целой переменной Current, то задавать позицию диаграммы в случае, если используются значения минимальной и максимальной позиции по умолчанию (т.е. 0 и 100), можно операторами:

```
Gaugel->SetValue(100 * Current / Count);
```

Можно поступать иначе: задать сначала значение максимальной величины равным Count, а затем в ходе процесса задавать позицию равной Current. Например:

```
Gaugel->SetRange(Count);
Gaugel->SetValue(Current);
```

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

Файл: *Unit1.cpp*

```
#include "stack.h"
...
Stack stack;
stack_projFrame::stack_projFrame(wxWindow* parent,wxWindowID id)
{
...
stack.setWxGrid(Grid1);
...
}
void stack_projFrame::OnPopbntClick(wxCommandEvent& event)
{
    if (!stack.isEmpty())
    {
        int val = stack.pop();
        wxString text;
        text << val;
        TextCtrl1->SetValue(text);
    }
    else
    {
        wxMessageBox(_("Stack is empty!!"));
    }
}
void stack_projFrame::OnPushbtnClick(wxCommandEvent& event)
{
    wxString text = TextCtrl1->GetValue();
    int val = wxAtoi(text);
    stack.push(val);
}
```

Файл: *stack.h*

```
#ifndef STACK_H
#define STACK_H
#include <wx/grid.h>
const int stacksize = 10;
const int EMPTY = -1;
class Stack
{
     int arr[stacksize]; // Массив для хранения данных
     int top; // Вершина стека
     wxGrid *wxg; // Указатель на StringGrid
public:
     Stack();  // Конструктор
     void push(int c);   // Добавление элемента
     int pop();      // Выталкивание элемента
     void clear();   // Очистка стека
     bool isEmpty(); // Проверка на наличие элементов в стеке
     bool isFull();  // Проверка на заполнение всего стека
     int getCount(); // Количество элементов в стеке
     void setWxGrid(wxGrid* _wxg);
};
#endif // STACK_H
```

Файл: *stack.cpp*

```
#include "stack.h"
Stack::Stack()
{
	top = EMPTY; // Изначально стек пуст
	wxg = NULL;
}
void Stack::clear()
{
	top = EMPTY;
}
bool Stack::isEmpty()
{
	return top == EMPTY;
}
bool Stack::isFull()
{
	return top == stacksize - 1;
}
int Stack::getCount()
{
	return top + 1; // Количество элементов в стеке
}
void Stack::push(int c)
{
	if(!isFull())
	{
		top++;
		arr[top] = c;
		if (wxg)
		{
			wxString s;
			s << c;
			wxg->InsertRows();
			wxg->SetCellValue(0,0,s);
		}
	}
}
int Stack::pop()
{
	if(!isEmpty())
	{
		top--;
		if (wxg) wxg->DeleteRows();
		return arr[top + 1];
	}
	else // Нечего извлекать
		return 0;
}
void Stack::setWxGrid(wxGrid* _wxg)
{
	wxg = _wxg;
}
```