Практическое занятие 16
«Динамические структуры данных. Стек»

Стек – это структура данных, в которой получить доступ к элементам можно лишь с одного конца, называемого **вершиной стека**; иначе говоря: стек – структура данных типа «список», функционирующая по принципу LIFO (last in — first out, «последним пришёл — первым вышел»). Графически его удобно изобразить в виде вертикального списка (см. рис. 1), например, стопки книг, где чтобы воспользоваться одной из них, и не нарушить установленный порядок, нужно поднять все те книги, что лежат выше нее, а положить книгу можно лишь поверх всех остальных.

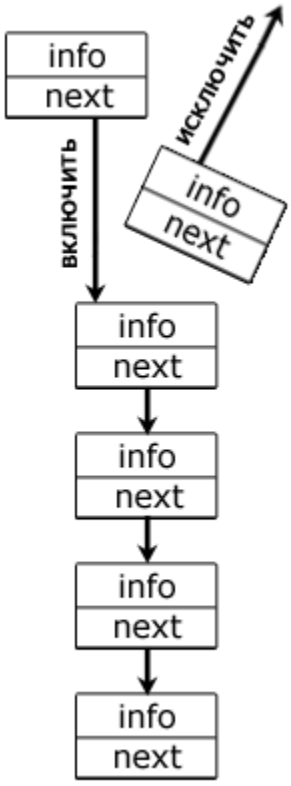

Рисунок 1. Графическое изображение стека

Операции над элементами происходят строго с одного конца: для включения нужного элемента в пятую по счету ячейку необходимо исключить тот элемент, который занимает эту позицию. Если бы было, например 6 элементов, а вставить конкретный элемент требовалось также в пятую ячейку, то исключить бы пришлось уже два элемента.

Стек чаще всего реализуется на основе обычных массивов, односвязных и двусвязных списков. В зависимости от конкретных условий, выбирается одна из этих структур данных.

Основными операциями над стеками являются:

- добавление элемента;
- удаление элемента;

- чтение верхнего элемента.

В языках программирования эти три операции, обычно дополняются и некоторыми другими.

Программа реализации стека на базе статического массива

В программе реализованы следующие операции, оформленные в виде функций:

- **Creation()** – создание пустого стека;
- **Full()** – проверка стека на пустоту;
- **Add()** – добавление элемента;
- **Delete()** – удаление элемента;
- **Top()** – вывод верхнего элемента;
- **Size()** – вывод размера стека.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
const int n=3;
struct Stack
{
int A[n];
int count;
};
//создание стека
void Creation(Stack *p)
{ p->count=0; }
//проверка стека на пустоту
int Full(Stack *p)
{
if (p->count==0) return 1;
else if (p->count==n) return -1;
else return 0;
}
//добавление элемента
void Add(Stack *p)
{
int value;
cout<<"Введите элемент > "; cin>>value;
p->A[p->count]=value;
p->count++;
}
```

```
 //удаление элемента
void Delete(Stack *p)
{ p->count--; }
 //вывод верхнего элемента
int Top(Stack *p)
{ return p->A[p->count-1]; }
 //размер стека
int Size(Stack *p)
{ return p->count; }
 //главная функция
void main()
{
setlocale(LC_ALL,"Russian");
Stack s;
Creation(&s);
char number;
do
{
cout<<"1. Добавить элемент"<<endl;
cout<<"2. Удалить элемент"<<endl;
cout<<"3. Вывести верхний элемент"<<endl;
cout<<"4. Узнать размер стека"<<endl;
cout<<"0. Выйти"<<endl;
cout<<"Номер команды > "; cin>>number;
switch (number)
{
case '1':
if (Full(&s)==-1) cout<<endl<<"Стек заполнен\n\n";
else
{
Add(&s);
cout<<endl<<"Элемент добавлен в стек\n\n";
} break;
//-------------------------------------------------
case '2':
if (Full(&s)==1) cout<<endl<<"Стек пуст\n\n";
else
{
Delete(&s);
cout<<endl<<"Элемент удален из стека\n\n";
} break;
//-------------------------------------------------
case '3':
if (Full(&s)==1) cout<<endl<<"Стек пуст\n\n";
else cout<<"\nВерхний элемент: "<<Top(&s)<<"\n\n";
break;
//-------------------------------------------------
case '4':
if (Full(&s)==1) cout<<endl<<"Стек пуст\n\n";
else cout<<"\nРазмер стека: "<<Size(&s)<<"\n\n";
break;
//-------------------------------------------------
case '0': break;
default: cout<<endl<<"Команда не определена\n\n";
break;
}
} while(number!='0');

system("pause");
}
```

Как видно, часто встречающимся элементом программы является поле count структуры stack. Оно исполняет роль указателя на «голову» стека. Например, для удаления элемента достаточно сдвинуть указатель на одну ячейку назад.

Программа реализации стека с использованием библиотеки шаблонов STL

Многие языки программирования располагают встроенными средствами организации и обработки стеков, в том числе C++.

Список функций C++ для работы со стеком:

- **push()** – добавить элемент;
- **pop()** – удалить элемент;
- **top()** – получить верхний элемент;
- **size()** – размер стека;
- **empty()** – проверить стек на наличие элементов.

Данные функции входят в стандартную библиотеку шаблонов C++ STL, а именно в контейнер **stack**. Использование стековых операций не требует их описания в программе, т.е. stack здесь предоставляет набор стандартных функций. Для начала работы со стеком в программе необходимо подключить библиотеку stack:

#include <stack>

и в функции описать стек:

stack <тип данных> имя стека;

Обратите внимание, что скобки, обособляющие тип данных, указываются здесь не как подчеркивающие общую форму записи, а как обязательные при описании стека.

```cpp
#include "stdafx.h"
#include <iostream>
#include <stack>
using namespace std;
//главная функция
void main()
{
setlocale(LC_ALL,"Russian");
stack <int> S; //создание стека S типа int
char number; int value;
do
{
cout<<"1. Добавить элемент"<<endl;
cout<<"2. Удалить элемент"<<endl;
cout<<"3. Получить верхний элемент"<<endl;
cout<<"4. Узнать размер стека"<<endl;
cout<<"0. Выйти"<<endl;
cout<<"Номер команды > "; cin>>number;
switch (number)
{
case '1': //добавление элемента
cout<<"Значение > "; cin>>value;
S.push(value); cout<<endl<<"Элемент добавлен в стек\n\n";
break;
//------------------------------------------------
case '2': //удаление элемента
if (S.empty()==true) cout<<"\nСтек пуст\n\n";
else
{
S.pop(); cout<<endl<<"Элемент удален из стека\n\n";
} break;
//------------------------------------------------
case '3': //вывод верхнего элемента
if (S.empty()==true) cout<<"\nСтек пуст\n\n";
else cout<<"\nВерхний элемент стека: "<<S.top()<<"\n\n";
break;

//------------------------------------------------
case '4': //вывод размера стека
if (S.empty()==true) cout<<"\nСтек пуст\n\n";
else cout<<"\nРазмер стека: "<<S.size()<<"\n\n";
break;
//------------------------------------------------
case '0': break; //выход
default: cout<<endl<<"Команда не определенная\n\n";
break;
}
} while(number!='0');

system("pause");
}
```

Индивидуальные задания

Создать стек с числами в диапазоне от –50 *до* +50. *После создания стека выполнить индивидуальное задание. В конце работы все стеки должны быть удалены.*

1. Преобразовать стек в два стека. Первый должен содержать только положительные числа, а второй – отрицательные.
2. Создать новый стек, содержащий только четные числа из первого стека.
3. Создать новый стек, содержащий только те числа из первого стека, которые больше среднего значения всех элементов первого стека.
4. Создать новый стек, содержащий только положительные числа из первого стека.
5. Подсчитать, сколько элементов стека имеют значение, которое превышает среднее значение всех элементов стека.
6. Найти максимальное и минимальное значения стека.
7. Определить, сколько элементов стека, начиная с вершины, находится до элемента с максимальным значением.
8. Определить, сколько элементов стека, начиная от вершины, находится до элемента с минимальным значением.
9. Определить, сколько элементов стека находится между его минимальным и максимальным элементами.
10. Определить, сколько элементов стека имеют значение меньше среднего значения всех элементов стека.
11. Создать новый стек, содержащий только числа, большие среднего значения всех элементов первого стека.
12. Преобразовать стек в два стека. В первый поместить все четные, а во второй – все нечетные числа.
13. Создать новый стек, порядок следования элементов в котором обратный относительно первого стека.
14. Создать новый стек, в который поместить каждый третий элемент первого стека.
15. Создать новый стек, в который поместить элементы, лежащие во второй половине первого стека.