ТЕМА 6.
ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ.
СТЕКИ

6.1. Понятие списка, стека и очереди

6.2. Понятие рекурсивных данных и стеков

```
struct TInf
{
        // Набор полей структуры
};

struct TNode
{
TInf inf;    // Информационная часть структуры
TNode* a;   // Адресная часть структуры
};
```

```
struct TNode
{
int  inf;          // Информационная часть структуры
TNode* a;    // Адресная часть структуры
};
```

6.3. Работа со стеком

Пример 6.1. Работа со стеком.

```
#include <iostream>
using namespace std;

struct TNode  // Описание элемента стека
{
  int inf;    // Информационная часть структуры
  TNode* a;   // Адресная часть структуры
};
```

```
struct stack // Структура для работы со стеком
{
TNode* top=nullptr; // Указатель на вершину стека
  int size=0;        // Количество элементов стека

  // Проверка наличия элементов в стеке
    bool empty()  {
        if (top) return false;
        else return true;
    }
```

```
// Добавление элемента в стек
    void push(int inf) {
        TNode* spt = new TNode;
        spt->inf = inf;
        spt->a = top;
        top = spt;
        size++;
    }
```

```
//  Удаление элемента из стека
void pop() {
    TNode *spt  = top;
    top = top->a;
    delete spt;
    size--;
}
```

```
// Вывод стека на экран
    void print() {
      TNode* spt = top;
      while (spt != nullptr) {
          cout << spt->inf << " ";
          spt = spt->a;
        }
      }
```

```
// Поиск элемента с заданным ключем
      TNode* search(int x) {
       if (!top) return nullptr;
          TNode* spt = top;
while (spt->inf != x && spt->a != nullptr) spt = spt->a;
        if (spt->inf == x) return spt;
        else return nullptr;
      }
```

```
// Поиск предыдущего эл-та (исключая первый)
      TNode* searchp(int x) {
        if (!top || !top->a) return nullptr;
          TNode* spt = top;
while (spt->a->inf != x && spt->a->a != nullptr)
                                                spt = spt->a;
          if (spt->a->inf == x) return spt;
          return nullptr;
      }
```

```
// Удаление элемента, с заданным ключём
        void del(int x) {
          if (!top) return;
          if (top->inf == x) pop();
           TNode* spt, * spp;
            spp = searchp(x);
            spt = spp->a;
            spp->a = spp->a->a;
        delete spt;
        }
```

```
// Обмен следующих за указанным элементов
        void exchange(TNode* sp) {
            TNode* spt;
            spt = sp->a->a;
            sp->a->a = spt->a;
            spt->a = sp->a;
            sp->a = spt;
        }
} ; // Конец объявления структуры
```

```
int main() {
  stack s;
s.push(4); s.push(2); s.push(1); s.push(6); s.push(9);
                                  s.print(); // Выводит: 9 6 1 2 4
   TNode* d1 = s.search(1);
                       cout << d1->inf;          // Выводит: 1
   TNode* d2 = s.searchp(1);
                       cout << d2->inf;          // Выводит: 2
    s.exchange(d2);    s.print(); // Выводит: 9 6 2 1 4
    s.del(6);                     s.print(); // Выводит: 9 2 1 4
 while (!s.empty()) s.pop();
 if (s.empty()) cout << "Stack is empty";
    return 0;   }
```