LANČANE LISTE

Čvor statičke dvostruko ulančane liste

```
template <class T>
class SDNode
{
public:
      T info;
      int prev;
      int next;

   SDNode() { prev = 0; next = 0; };
   SDNode(T i) { info = i; prev = 0; next = 0; };
   SDNode(T i, int p, int n) {
                        info = i; prev = p; next = n;};
   ~SDNode() { };
   T print() {return info;};
   bool isEqual(T el) {return el == info; };
};
```

Statička dvostruko ulančana lista

```
template <class T>
SDLList<T>::SDLList()
{
   size = 0;

   head = tail = 0;
   lrmp = 0;
   data = NULL;
}

template <class T>
SDLList<T>::~SDLList()
{
   if (data != NULL) {
         delete[] data;
   }
}
```

```
template <class T>
SDLList<T>::SDLList(int n)
{
   size = n;

   head = tail = 0;
   lrmp = 1;
   data = new SDNode<T>[size+1];

   int i;
   for (i=1; i<size; i++) {
         data[i].next = i+1;
   }
   data[size].next = 0;
}
```

Statička dvostruko ulančana lista

```
template <class T>
void SDLList<T>::printAll(bool bFromHead) {
  if (bFromHead) {
      for (int tmp=head; tmp!=0;
                  tmp=data[tmp].next)
            cout << data[tmp].print() << " ";
      cout << endl;
  } else {
      for (int tmp=tail; tmp!=0;
                  tmp=data[tmp].prev)
            cout << data[tmp].print() << " ";
      cout << endl;
  }
}
```

Statička dvostruko ulančana lista

```
template <class T>
void SDLList<T>::addToHead(T el) {
   if (lrmp == 0) {
         return;
   }
   int tmp = lrmp;
   lrmp = data[lrmp].next;
   data[tmp].info = el;
   data[tmp].prev = 0;
   data[tmp].next = head;

   if (head == 0) {
         tail = head = tmp;
   } else {
         data[head].prev = tmp;
         head = tmp;
   }
}
```

Statička dvostruko ulančana lista

```
template <class T>
void SDLList<T>::addToTail(T el) {
  if (lrmp == 0) {
      return;
  }
  int tmp = lrmp;
  lrmp = data[lrmp].next;
  data[tmp].info = el;
  data[tmp].prev = tail;
  data[tmp].next = 0;
  if (tail == 0) {
      tail = head = tmp;
  } else {
      data[tail].next = tmp;
      tail = tmp;
  }
}
```

Statička dvostruko ulančana lista

```
template <class T>
T SDLList<T>::deleteFromHead() {
   if (head == 0)
         return T();
   int tmp = head;
   if (head == tail)
         head = tail = 0;
   else {
         head = data[head].next;
         data[head].prev = 0;
   }

   T el = data[tmp].info;

   data[tmp].prev = 0;
   data[tmp].next = lrmp;
   lrmp = tmp;

   return el;
}
```

Statička dvostruko ulančana lista

```
template <class T>
T SDLList<T>::deleteFromTail() {
   if (head == 0)
         return T();
   int tmp = tail;
   if (head == tail)
         head = tail = 0;
   else {
         tail = data[tail].prev;
         data[tail].next = 0;
   }

   T el = data[tmp].info;

   data[tmp].prev = 0;
   data[tmp].next = lrmp;
   lrmp = tmp;

   return el;
}
```

Statička dvostruko ulančana lista

```
template <class T>
void SDLList<T>::deleteEl(T el) {
   int tmp;
   for (tmp = head; tmp != 0 && data[tmp].info != el;
                                          tmp = data[tmp].next);
   if (tmp != 0) {
        if (tmp == head)
                head = data[head].next;
        else
                data[data[tmp].prev].next = data[tmp].next;
        if (tmp == tail)
                tail = data[tail].prev;
        else
                data[data[tmp].next].prev = data[tmp].prev;

        data[tmp].prev = 0;
        data[tmp].next = lrmp;
        lrmp = tmp;
   }
}
```