Свързан списък

доц. д-р Нора Ангелова

Свързан списък

Логическо представяне

- Съставна хомогенна линейна структура от данни - крайна редица от елементи от един и същ тип

Достъп

- възможен е достъп до всеки елемент в редицата (последователен).

Операции

- добавяне, премахване и промяна на елемент - допустими са на произволно място в редицата.
- обхождане – посещение на всеки от елементите точно по веднъж (използване на итератор)
- търсене на елемент

Свързан списък

Физическо представяне

- последователно (непрепоръчително) – защо? Каква би била сложността на операциите?
- свързано

Свързан списък

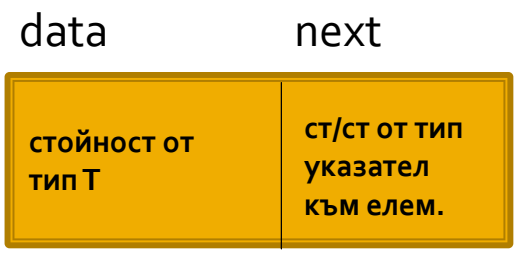
Типове

- свързан списък с една връзка
- свързан списък с две връзки
- цикличен свърза списък
- свързан списък с прескачане

Свързан списък с една връзка

- Реализация на свързан списък с итератор
  - представяне на елемент

```
template <typename T>

struct LinkedListElement {

   T data;

   LinkedListElement<T>* next;

};
```



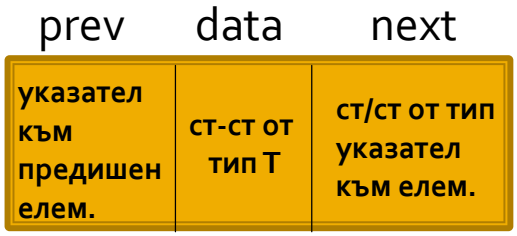
  - реализация на класа (вътрешно представяне)

```
public:
  typedef LListIterator<T> I;

private:
  typedef LinkedListElement<T> LE;
  LE *front, *back;
```

Свързан списък с две връзки

- Реализация на двойносвързан списък с итератор
  - представяне на елемент

```
template <typename T>

struct DoubleLinkedListElement {

   T data;

   DoubleLinkedListElement<T>* next;

};
```



  - реализация на класа (вътрешно представяне)

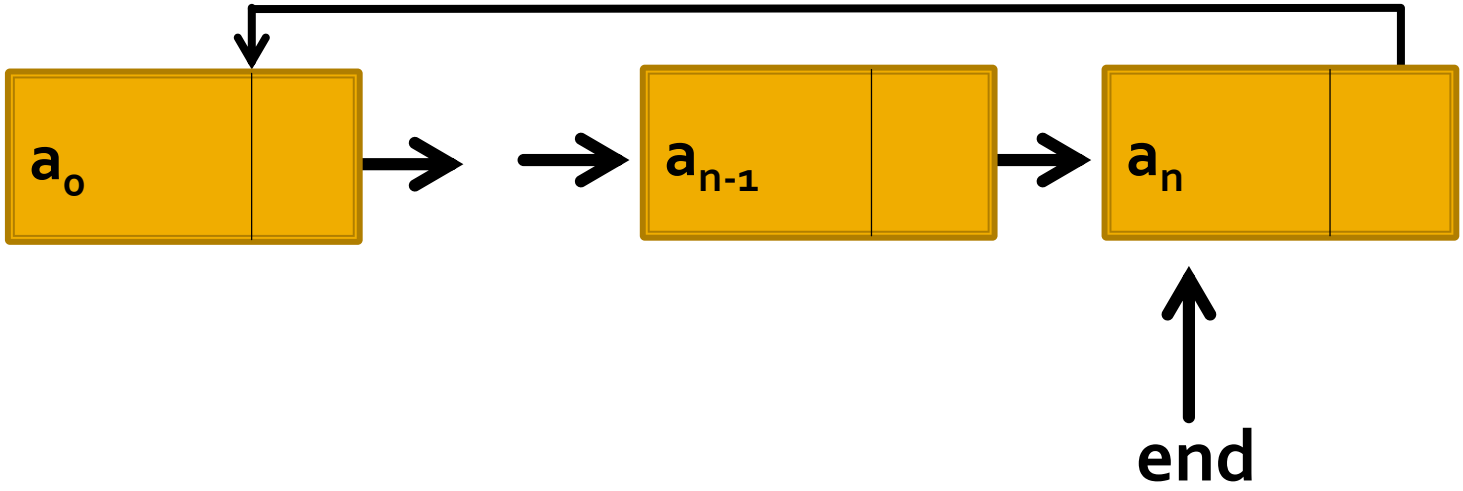
```
public:
  typedef LListIterator<T> I;

private:
  typedef LinkedListElement<T> LE;
  LE *front, *back;
```

Цикличен свързан списък

- хомогенна линейна структура от данни
- последният елемент на свързания списък съдържа връзка към първия елемент на свързания списък
- обхождане на елементите?
- посещаването на елементите може да се случва многократно

Цикличен свързан списък

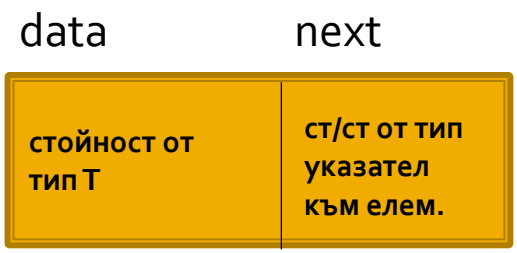
- Графично представяне
- Възможна е реализация с един указател
- По-удобно е той да сочи последния елемент на свързания списък



Цикличен свързан списък

- Реализация на цикличен свързан списък
  - представяне на елемент

```
template <typename T>
struct CListElement {
   T data;
   LinkedListElement<T>* next;
};
```

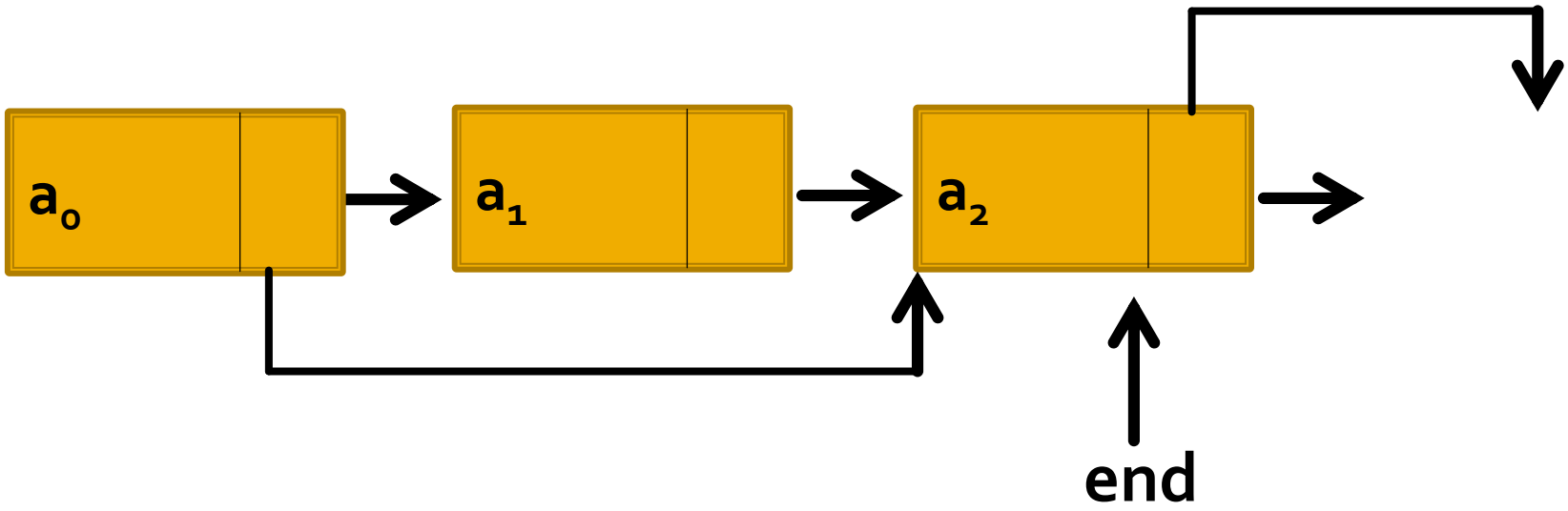


Свързан списък с прескачане

- хомогенна линейна структура от данни
- позволява да се прескачат елементи
- дава възможност за изграждане на нива от елементите на даден свързан списък
- всяко ниво съдържа по-малко елементи и няма нови такива

Свързан списък с прескачане

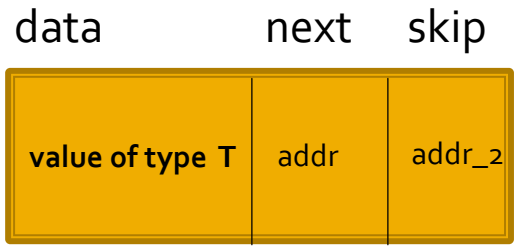
- графично представяне
- възможна е реализация с един допълнителен указател към следващ елемент



Свързан списък с прескачане

- Реализация на двойносвързан списък с итератор
  - представяне на елемент

```
template <typename T>
struct SkipListElement {
   T data;
   SkipListElement<T> *next, *skip;
};
```



STL (Списък)

`std::list<T>`

`#include <list>`

** Реализацията на std::list в STL е двусвързана.*

Интерфейс:

- front(), back() - първи и последен елемент
- begin(), end() - итератори към началото и края
- rbegin(), rend() - итератори за обратно обхождане
- push_front(), push_back() - вмъкване в началото/края
- pop_front(), pop_back() - изтриване от началото/края
- insert(), erase() - вмъкване/изтриване на позиция
- splice() - прехвърляне на елементи от един списък в друг
- remove(), remove_if() - филтриране по стойност/предикат
- merge() - сливане на подредени списъци
- sort() - сортиране на списък (на място)
- reverse() - обръщане на списък

- ==, !=, <=, >= - лексикорафско сравнение на два списъка

Край