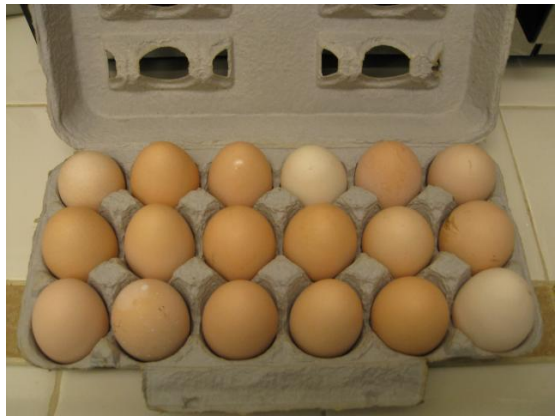
Редици

Трифон Трифонов

Структури от данни и програмиране,
спец. Компютърни науки, 2 поток, 2015/16 г.

23 октомври 2015 г.



АТД: Масив

Последователност от елементи от еднакъв вид, които могат да бъдат избирани по номер (индекс).

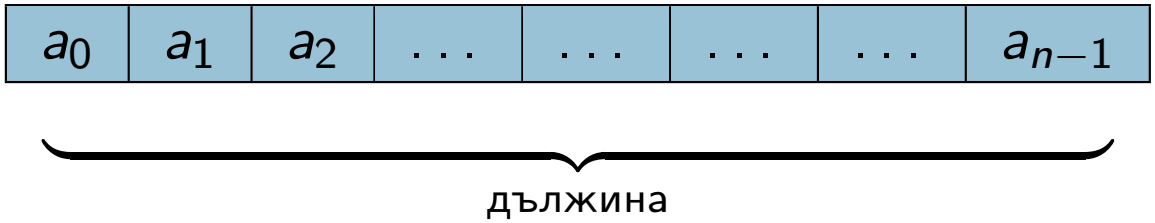
Операции

- `create(n)` — създаване на масив със зададена големина
- `get(i)` — получаване на елемент с индекс `i`
- `set(i,x)` — задаване на стойност `x` на елемента с индекс `i`
- `size` — дължина на масива

Свойства на операциите

- `a.set(i,x).get(i)` = `x`
- `a.set(i,x).get(j)` = `a.get(j)`, ако `i` $\neq$ `j`
- `create(n).size` = `n`

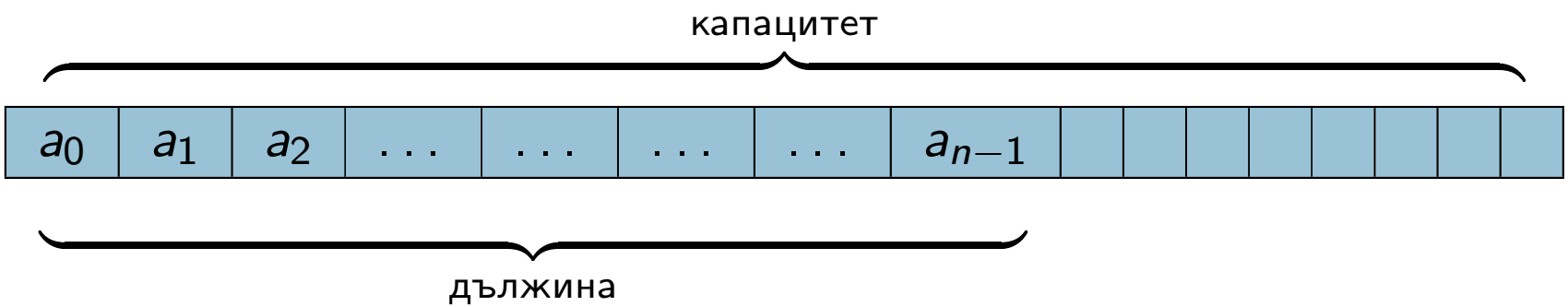
Статично представяне



Реализация: масив във C++.

Пример: `int a[10];`

Динамично представяне



Реализация: `std::vector`.

std::vector<T>

Реализация на динамичен масив

- `vector(n)` — създава вектор с дължина `n`
- `size` — дължина на вектора
- `capacity` — капацитет на вектора
- `[i]`, `at(i)` — достъп до елемент на индекс `i`
- `front()`, `back()` — достъп до първи и последен елемент
- `push_back(x)` — добавяне на елемента `x` в края
- `pop_back()` — изтриване на последния елемент
- `insert(...)` — вмъкване на елементи на произволна позиция
- `erase(...)` — изтриване на елементи на произволна позиция
- `==,!=,<,>,<=,>=` — лексикорафско сравнение на два вектора

std::vector<T>

Реализация на динамичен масив

- `vector(n)` — създава вектор с дължина `n`
- `size` — дължина на вектора
- `capacity` — капацитет на вектора
- `[i]`, `at(i)` — достъп до елемент на индекс `i`
- `front()`, `back()` — достъп до първи и последен елемент
- `push_back(x)` — добавяне на елемента `x` в края
- `pop_back()` — изтриване на последния елемент
- `insert(...)` — вмъкване на елементи на произволна позиция
- `erase(...)` — изтриване на елементи на произволна позиция
- `==`, `!=`, `<`, `>`, `<=`, `>=` — лексикорафско сравнение на два вектора

Специализация `vector<bool>`: реализирана чрез битови масиви

std::string

Реализация на низ (динамична редица от символи)

- Всички методи на `std::vector<char>`
  - но не го наследява!
- Методите са съвместими с `char*`
- `replace(...)` — подмяна на символи на произволна позиции
- `+,+=,append(...)` — конкатенация на низове
- `<<, >>` — операции за вход и изход
- `c_str()` — конвертиране към стандартен C++ низ
- `find(...), rfind(...)` — търсене на първо/последно срещане
- `find_first_of(...)` — първо срещане на символ от друг низ
- `substr(...)` — извличане на подниз
- `compare(...)` — сравнение с друг низ
- `copy(...)` — копиране на символи от C++ низ

АТД: Наредена двойка

Двойка от елементи от потенциално различен тип, в която редът има значение.

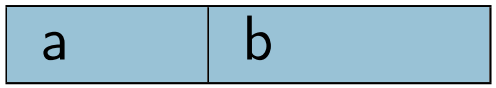
Операции

- `create(a,b)` — създава двойка от елементите `a` и `b`
- `p.first` — връща първия елемент на двойката
- `p.second` — връща втория елемент на двойката

Свойства на операциите

- `create(a,b).first` = `a`
- `create(a,b).second` = `b`
- `create(p.first,p.second)` = `p`

Физическо представяне



Реализации:

- `struct Pair{ int first; char second; };`
- `std::pair<T,U>`

std::pair

Реализация на наредена двойка

- `pair(x,y)` — създаване на двойка (`x`,`y`)
- `first` — първи елемент
- `second` — втори елемент
- `==,!=,<,>,<=,>=` — лексикорафско сравнение на две двойки

АТД: Кортеж

Редица от фиксиран брой елементи от потенциално различен тип, в която редът има значение.

Операции

- `create(...)` — създаване на кортеж по единични елементи
- `get(i)` — получаване на елемент с индекс/име `i`
- `set(i,x)` — задаване на стойност `x` на елемента с индекс/име `i`

Свойства на операциите

- $\texttt{create}(a_1,\ldots,a_i,\ldots,a_n).\texttt{get}(i) = a_i$
- $\texttt{t.set}(i,x).\texttt{get}(i) = x$
- $\texttt{t.set}(i,x).\texttt{get}(j) = \texttt{a.get}(j)$, ако $i \neq j$