IDT, Přednáška 7

Libor Váša

Katedra informatiky a výpočetní techniky, Západočeská univerzita v Plzni

18. 3. 2024

ADT Fronta

Fronta (Queue)

- abstraktní datová struktura
- podobá se zásobníku
- místo posledního umožňuje vybrat/odstranit první prvek
  - ten který se v kolekci nachází nejdelší dobu
- přirozená reprezentace pro odpovídající situaci:
  - vyřizování požadavků v tom pořadí, v jakém vznikly
- FIFO: First In, First Out

Fronta (Queue)

Implementace ADT Fronta

následující operace ideálně v konstantním čase:

- přidání prvku na konec
- vybrání prvku na začátku
- odstranění prvku na začátku
- zjištění, zda je prázdná

Implementace ADT Fronta

následující operace ideálně v konstantním čase:

- přidání prvku na konec
- vybrání prvku na začátku
- odstranění prvku na začátku
- zjištění, zda je prázdná

možné implementace

- spojová struktura
- pole

Rozhraní Fronty

```
interface IQueue<T> {
  void Add(T e);
  T Get();
  void RemoveFirst();
  bool IsEmpty();
}
```

Implementace fronty spojovou strukturou

spojovací dílek

```
class Link <T>{
  T data;
  Link next;
}
```

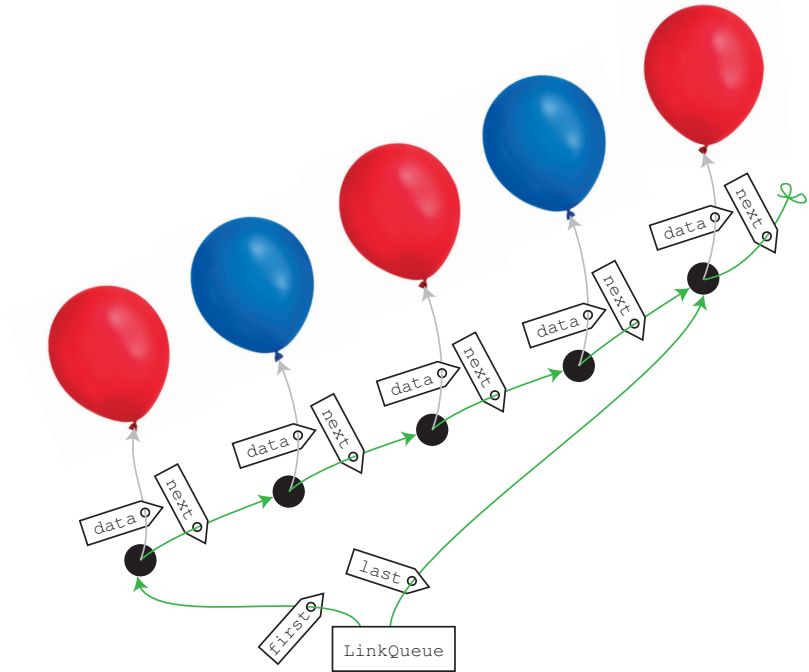
Implementace fronty spojovou strukturou

spojovací dílek

```
class Link <T>{
  T data;
  Link next;
}
```

třída si drží v referenčních proměnných odkazy na první i poslední prvek

```
class LinkQueue<T> : IQueue<T>{
  public Link<T> first;
  public Link<T> last;
  ...
}
```


data
next
data
next
data
next
data
next
data
next
first
last
LinkQueue

Přidání položky do fronty

podobné jako u zásobníku

```
void Add(T e){
  Link nl = new Link();
  nl.data = e;
```

Přidání položky do fronty

podobné jako u zásobníku

```
void Add(T e){
  Link nl = new Link();
  nl.data = e;
  if (first == null){
    first = nl;
    last = nl;
  }
```

Přidání položky do fronty

podobné jako u zásobníku

```
void Add(T e){
  Link nl = new Link();
  nl.data = e;
  if (first == null){
    first = nl;
    last = nl;
  }
  else{
    last.next = nl;
    last = nl;
  }
}
```

Vybrání, odstranění

Vybrání prvního prvku

```
T Get(){
  if (first!=null)
    return first.data;
  else throw new Exception();
}
```

Vybrání, odstranění

Vybrání prvního prvku

```
T Get(){
  if (first!=null)
    return first.data;
  else throw new Exception();
}
```

Odstranění prvního prvku

```
void RemoveFirst(){
  if (first!=null)
    first = first.next;
  else throw new Exception();
}
```

Zjištění, zda je prázdná

Vybrání prvního prvku

```
bool IsEmpty(){
  return (first==null);
}
```

Složitosti

- všechny operace je možné provést v čase $\Theta(1)$
- potenciální problém s pamětí
  - je možné, že velikost spojovacího dílku v paměti je větší než velikost samotných dat

Implementace ADT fronta polem

- podobné jako u zásobníku
- pro odstranění není vhodné přesouvat prvky
- struktura si drží
  - index prvního prvku
  - počet obsazených indexů
- postupně se uvolňuje místo na začátku pole, při přidávání prvků je možné ho využít

Implementace ADT Fronta polem

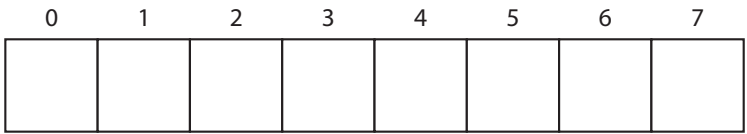
```
class ArrayQueue<T> : IQueue<T>{
  T[] array;
  int first;
  int count;
  ...
}
```

Použití

```
ArrayQueue queue = new ArrayQueue();
queue.Add(15);
queue.Add(30);
queue.Add(12);
queue.Add(14);
queue.Add(-1);
queue.Add(0);
queue.Add(42);
queue.RemoveFirst();
queue.Add(33);
queue.Add(85);
queue.Add(1);
```

Ilustrace

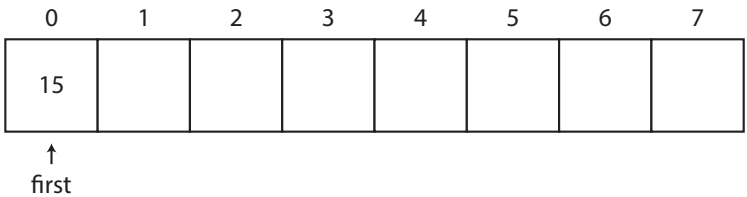
count: 0



Operace:

Ilustrace

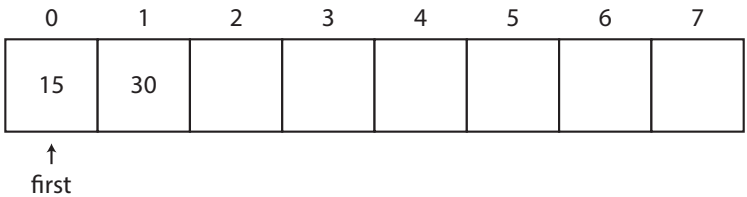
count: 1



Operace: add(15);

Ilustrace

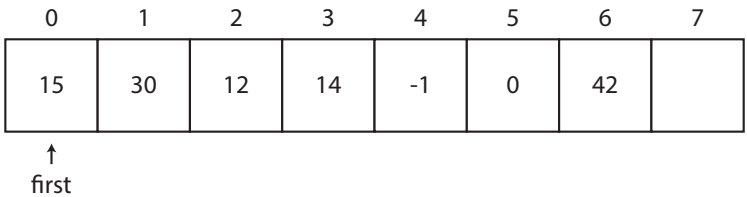
count: 2



Operace: add(15); add(30);

Ilustrace

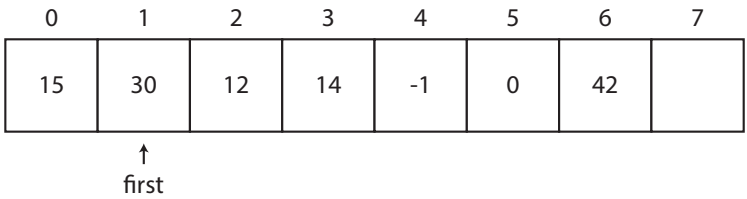
count: 7



Operace: add(15); add(30); add(12); add(14); add(-1); add(0); add(42)

Ilustrace

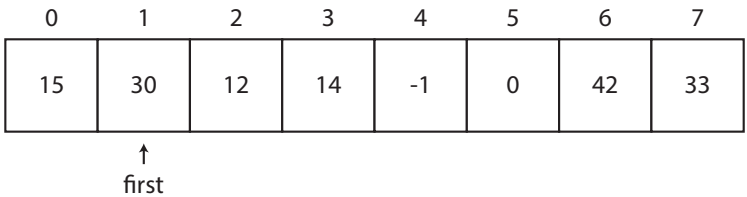
count: 6



Operace: add(15); add(30); add(12); add(14); add(-1); add(0); add(42); removeFirst();

Ilustrace

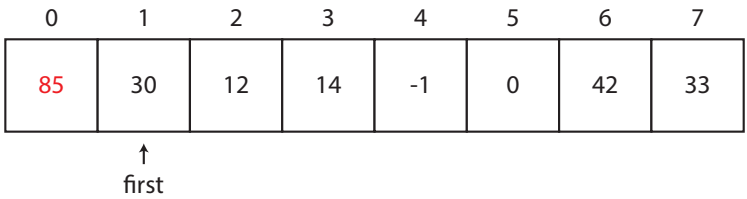
count: 7



Operace: add(15); add(30); add(12); add(14); add(-1); add(0); add(42); removeFirst(); add(33);

Ilustrace

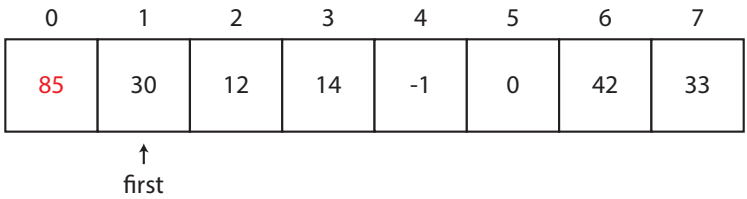
count: 8



Operace: add(15); add(30); add(12); add(14); add(-1); add(0); add(42); removeFirst(); add(33); add(85);

Ilustrace

count: 8



Operace: add(15); add(30); add(12); add(14); add(-1); add(0); add(42); removeFirst(); add(33); add(85); add(1) vede na zvětšení pole

Vybrání a odstranění prvku

Vybrání prvního prvku

```
T Get(){
  if (count>0)
    return(array[first]);
  else throw new Exception();
}
```

Odstranění prvního prvku

```
void RemoveFirst(){
  if (count>0){
    first = (first+1)%array.Length;
    count--;
  }
  else throw new Exception();
}
```

Přidání prvku

musí řešit, že se prvek nevejde

```
void Add(T e){
  if (count == array.Length)
    ExpandArray();
  array[(first + count)%array.Length] = e;
  count++;
}
```

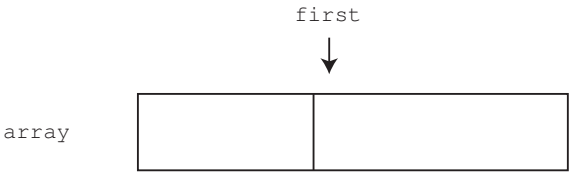
Přidání prvku

musí řešit, že se prvek nevejde

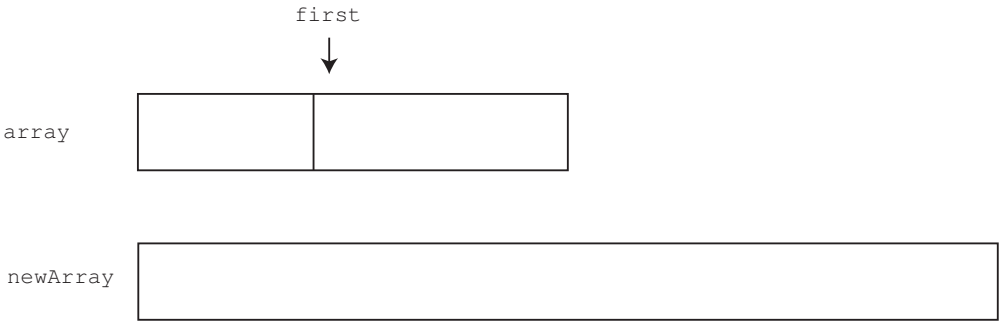
```
void Add(T e){
  if (count == array.Length)
    ExpandArray();
  array[(first + count)%array.Length] = e;
  count++;
}

void ExpandArray(){
  T[] newArray = new T[array.Length*2];
  for (int i = 0;i<array.Length;i++)
    newArray[i] = array[(first + i)%array.Length)];
  array = newArray;
  first = 0;
}
```

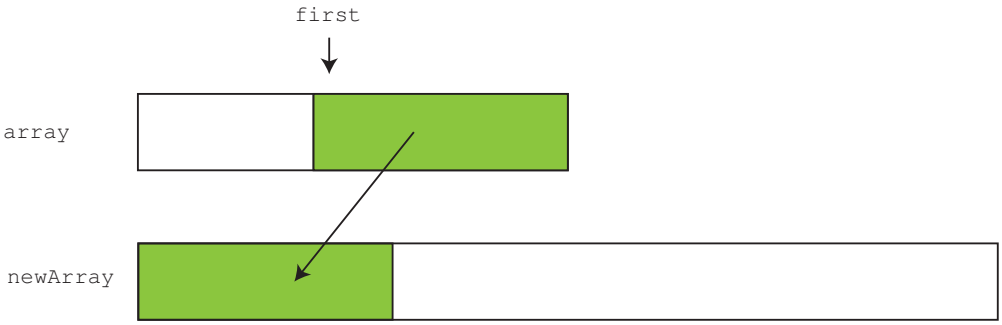
Ilustrace



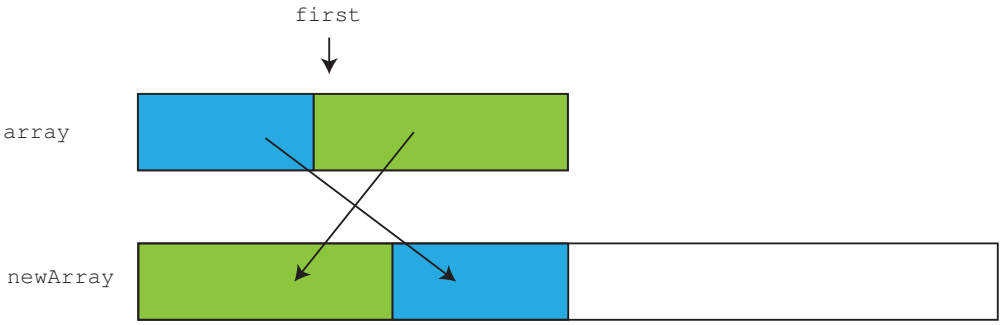
Ilustrace



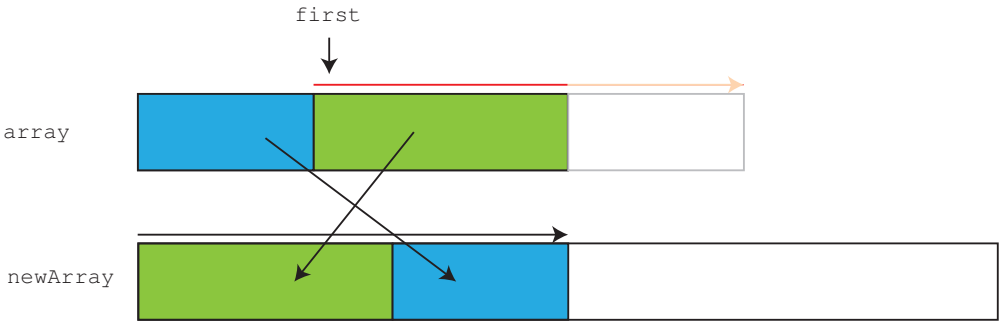
Ilustrace



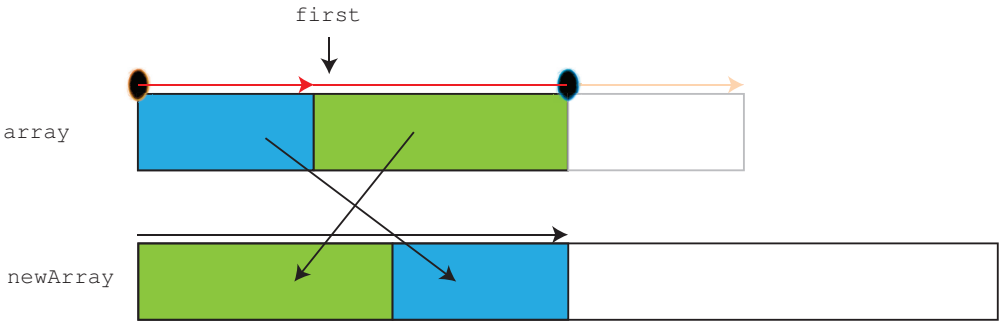
Ilustrace



Ilustrace



Ilustrace



Zjištění, zda je fronta prázdná

```
void IsEmpty(T e){
  return (count == 0)
}
```

Výpočetní složitost

- Vybrání/odstranění prvního prvku: $\Theta(1)$
- Přidání prvku na konec:
  - většinou $\Theta(1)$
  - někdy $\Theta(n)$: když expanduje pole
  - v průměru $\Theta(1)$
    - stejná agregovaná složitost jako v případě zásobníku

Obvyklé rozhraní fronty

- Přidání prvku na konec: `Enqueue(T e)`
- Vybrání + odstranění prvního prvku: `T Dequeue()`
- Vybrání prvního prvku `T Peek()`

```
interface IQueue<T> {
  Enqueue(T e);
  T Dequeue();
  T Peek();
  bool IsEmpty();
}
```