Алгоритмы и структуры данных

Лекция 9. Стек и очередь на списках

Антон Штанюк (к.т.н, доцент)

14 апреля 2022 г.

Нижегородский государственный технический университет им. Р.Е. Алексеева
Институт радиоэлектроники информационных технологий
Кафедра ”Компьютерные технологии в проектировании и производстве”

Содержание

Введение

В этой теме мы рассмотрим реализацию широко известных логических структур данных **стека** и **очереди** на связанных списках. В подобной реализации есть ряд преимуществ. Самое главное из них, - это неограниченные, определяемые только размером свободной памяти, возможности по увеличении числа элементов. Второе преимущество заключается в использовании даже сильно фрагментированной памяти, где разместить большой массив проблематично.

Реализация стека на списке

Реализация стека на списке

Для реализации стека на списке, нам потребуется взять за основу класс **LList**, введенный в прошлой теме и немного его модифицировать, удалив лишние операции. Необходимо оставить добавление элементов с одного конца списка и удаление с того же конца.

Реализация стека на списке

```
#include <cassert>

template<typename T>
class LListStack {
    struct ITEM {
        T data;
        ITEM * next;
    };
public:
    LListStack():head(nullptr){}
    ~LListStack();
    void push(const T&);
    T pop();
    void print() const;
private:
    LListStack::ITEM* create(const T&);
    ITEM *head;
};
```

Реализация стека на списке

В качестве программного интерфейса мы используем:

- **create()** - метод для создания нового звена.
- **push()** - метод добавления нового элемента в стек.
- **pop()** - метод извлечения элемента.

Деструктор будет нужен для удаления всех элементов стека.

Реализация стека на списке

```
template<typename T>
LListStack<T>::~LListStack() {
    while(head)
        pop();
}
```

При добавлении в стек данных создается новое звено

```
template<typename T>
typename LListStack<T>::ITEM* LListStack<T>::create(const T& data) {
   ITEM *item=new ITEM;
   item->data=data;
   item->next=nullptr;
   return item;
}
```

Реализация стека на списке

Непосредственно реализация метода **push**:

```
template<typename T>
void LListStack<T>::push(const T& data) {
    if(head) {
        ITEM *temp=create(data);
        temp—>next=head;
        head=temp;
    }
    else {
        head=create(data);
    }
}
```

Реализация стека на списке

При извлечении элемента из списка удаляется головной элемент. При отсутствии элементов возвращается нулевое значение (можно выбрасывать исключение)

```
template<typename T>
T LListStack<T>::pop() {
    if(head) {
        ITEM *temp=head->next;
        T data=head->data;
        delete head;
        head=temp;
        return data;
    }
    else {
        throw std::string("Empty!");
    }
}
```

Реализация стека на списке

Для того, чтобы можно было распечатывать стек, напишем метод **print**:

```
template<typename T>
void LListStack<T>::print() const {
    ITEM *temp=head;
    while(temp) {
        std::cout<<temp->data<<"␣";
        temp=temp->next;
    }
    std::cout<<std::endl;
}
```

Пример использования стека

```
#include "tlist.h"

int main() {
    int i;
    LListStack<int> stack;
    for(i=1;i<10;i++) {
        stack.push(i);
        stack.print();
    }
    while(stack.pop())
        stack.print();
    return 0;
}
```

Результат работы

```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
9 8 7 6 5 4 3 2 1
8 7 6 5 4 3 2 1
7 6 5 4 3 2 1
6 5 4 3 2 1
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

Реализация очереди на списке

Очередь на списке

Для реализации очереди на списке, нужно взять класс **LList** и оставить добавление элементов в хвост и удаление головного элемента.

Очередь на списке

```
template<typename T>
class LListQueue {
    struct ITEM {
        T data;
        ITEM * next;
    };
public:
    LListQueue():head(nullptr),tail(nullptr){}
    ~LListQueue();
    void push(const T&);
    T pop();
    void print() const;
private:
    LListQueue::ITEM* create(const T&);
    ITEM *head;
    ITEM *tail;
};
```

Очередь на списке

```
template<typename T>
typename LListQueue<T>::ITEM* LListQueue<T>::create(const T& data) {
   ITEM *item=new ITEM;
   item->data=data;
   item->next=nullptr;
   return item;
}
```

Очередь на списке

```
template<typename T>
LListQueue<T>::~LListQueue() {
    while(head)
      pop();
}
```

Очередь на списке

```
template<typename T>
void LListQueue<T>::push(const T& data) {
    if(tail && head) {
        tail->next=create(data);
        tail=tail->next;
    }
    else {
        head=create(data);
        tail=head;
    }
}
```

Очередь на списке

```
template<typename T>
T LListQueue<T>::pop() {
    if(head) {
        ITEM *temp=head->next;
        T data=head->data;
        delete head;
        head=temp;
        return data;
    }
    else {
        throw std::string("Empty!");
    }
}
```

Очередь на списке

```
template<typename T>
void LListQueue<T>::print() const {
    ITEM *temp=head;
    while(temp) {
        std::cout<<temp->data<<"␣";
        temp=temp->next;
    }
    std::cout<<std::endl;
}
```

Список литературы

Список литературы i

- Кормен Т.,Лейзерсон Ч., Ривест Р.
  Алгоритмы: построение и анализ
  МЦНМО, Москва, 2000
- **Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ.**
  2-е изд. — М.: «Вильямс», 2006
- **Википедия**
  Алгоритм
  http://ru.wikipedia.org/wiki/Алгоритм
- **Википедия**
  Список алгоритмов
  http://ru.wikipedia.org/wiki/Список_алгоритмов
- **Традиция**
  Задача коммивояжёра
  http://traditio.ru/wiki/Задача

Список литературы ii

- **Википедия**
  NP-полная задача
  http://ru.wikipedia.org/wiki/NP-полная
- **Серджвик Р.**
  Фундаментальные алгоритмы на C++. Части 1-4
  Diasoft,2001
- **Седжвик Р.**
  Фундаментальные алгоритмы на C. Анализ/Структуры данных/Сортировка/Поиск
  СПб.: ДиаСофтЮП, 2003
- **Седжвик Р.**
  Фундаментальные алгоритмы на C. Алгоритмы на графах
  СПб.: ДиаСофтЮП, 2003
- **Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы.**
  Издательский дом «Вильямс», 2000

Список литературы iii

- Кнут Д.
  Искусство программирования, том 1. Основные алгоритмы
  3-е изд. — М.: «Вильямс», 2006
- Кнут Д.
  Искусство программирования, том 2. Получисленные методы
  3-е изд. — М.: «Вильямс», 2007
- Кнут Д.
  Искусство программирования, том 3. Сортировка и поиск
  2-е изд. — М.: «Вильямс», 2007
- Кнут Д.
  Искусство программирования, том 4, выпуск 3. Генерация всех сочетаний и разбиений
  М.: «Вильямс», 2007

Список литературы iv

Кнут Д.
Искусство программирования, том 4, выпуск 4. Генерация всех деревьев. История комбинаторной генерации
М.: «Вильямс», 2007