Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

**РЕФЕРАТ**

По предмету “Основы алгоритмизации и программирования”

На тему: “Двусвязный список”

Выполнил:

Студент 1 курса 9 группы

Павлович Ян Андреевич

Преподаватель: Белодед Н.И.

2024, Минск

Содержание

1. Введение
2. Основная часть
   1. Двусвязный список: определение
   2. Немного про шаблоны классов в C++
   3. Общий вид двусвязного списка
   4. Создание двусвязного списка
   5. Добавление элемента
      1. Добавление элемента в пустой список
      2. Добавление элемента в список, содержащий элементы
   6. Преимущества и недостатки двусвязного списка
3. Заключение
4. Источники

**Введение**

Двусвязные списки предоставляют удобную альтернативу массивам и другим структурам данных благодаря своей способности хранить элементы последовательно и обеспечивать быстрый доступ к ним как в прямом, так и в обратном направлении. Это обеспечивает гибкость и эффективность при выполнении операций вставки, удаления и обхода элементов списка.

В данном реферате мы рассмотрим определение и структуру двусвязных списков, их реализацию с использованием шаблонных классов в языке программирования C++, а также их применение, преимущества и недостатки. Я решил использовать шаблоны классов в реферате, так как столкнулся с ними при реализации одной из программ. Поэтому, чтобы больше разобраться в односвязных и двусвязных списках именно с использованием шаблона классов было принято решение использовать их и в реферате.

**Двусвязный список: определение**

Двусвязный список - это особый вид динамической структуры данных, который состоит из узлов. Эти узлы уникальны тем, что каждый из них содержит ссылки на предыдущий и следующий узлы в списке, а также данные. В отличие от односвязного списка, где узлы имеют ссылку только на следующий узел, двусвязный список позволяет нам перемещаться в обоих направлениях. Это обеспечивает большую гибкость при проектировании алгоритмов.

Благодаря использованию шаблонных классов, двусвязные списки могут быть настроены так, чтобы хранить любой тип данных, что делает их универсальным инструментом для программистов.

**Немного про шаблоны классов в C++**

Для понимания реализации двусвязного списка с помощью шаблона классов нужно для начала понять, что это такое.Шаблонные классы в C++ представляют собой мощный инструмент, позволяющий создавать обобщенные структуры данных и функции, которые могут работать с различными типами данных, не завися от конкретного типа. Они являются ключевой особенностью языка C++, обеспечивая гибкость и переиспользование кода.

Основные черты шаблонных классов в C++:

**Обобщенность:** Шаблонные классы позволяют определить класс с параметром, который может быть заменен на конкретный тип данных при создании объекта класса. Например, **template<typename T>** определяет класс с типовым параметром **T**, который может представлять любой тип данных.

**Параметризация:** Параметры шаблона могут быть использованы в определении класса, методов и переменных, что позволяет создавать обобщенные решения, не привязанные к конкретным типам данных.

**Универсальность:** Шаблонные классы могут быть использованы для создания контейнеров, алгоритмов, функций и других структур данных, которые могут работать с различными типами данных, без необходимости переписывания кода для каждого типа.

Пример шаблонного класса:

template<typename T>

class Item {

private:

    // Приватные члены класса, которые необходимо описать

public:

    // Публичные члены класса, доступные извне

};

Публичные и приватные члены класса в языке программирования C++ определяют уровень доступа к членам класса и определяют, кто имеет право использовать или модифицировать эти члены.

**Публичные члены класса (public)**:

Публичные члены класса доступны извне класса, т.е. из других частей программы, включая функции и объекты других классов.

Они обычно используются для предоставления интерфейса класса, т.е. набора методов и операций, доступных для внешнего использования.

Публичные члены обычно документируются и представляются как функции и переменные, с которыми разработчик может взаимодействовать.

**Приватные члены класса (private)**:

Приватные члены класса доступны только внутри самого класса.

Они обычно используются для внутренней реализации класса и данных, которые должны быть скрыты от внешнего мира.

Приватные члены обычно не доступны для использования вне класса и могут быть доступны только через публичные методы класса.

**Общий вид двусвязного списка**

Двусвязный список состоит из узлов, каждый из которых содержит данные и два указателя:

* **data** – данные. Содержимым может быть любое значение любого типа;
* **next** – указатель на следующий элемент;
* **prev** – указатель на предыдущий элемент.

Основной элемент списка:

template ‹class T› //шаблонный класс, где T является параметром шаблона. То есть позволяет использовать один и тот же код для различных типов данных

struct Item //структура, которая представляет узел двусвязного списка

{

T data; //данные, хранящиеся в узле списка

Item<T>\* next; //указатель на следующий узел списка

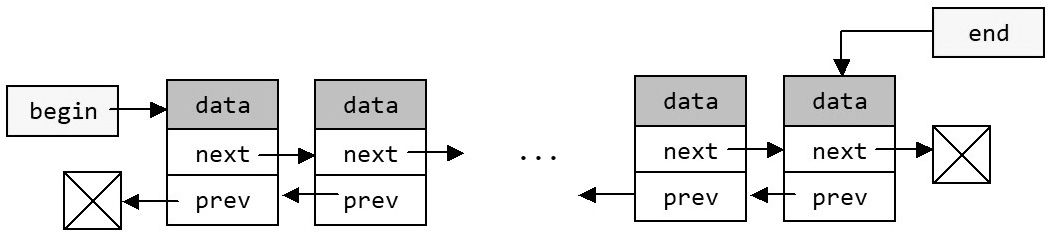
Item<T>\* prev; //указатель на предыдущий узел списка

};

В начале и конце списка могут находиться особые узлы, которые называются головой и хвостом соответственно. Голова списка содержит указатель только на следующий элемент, а хвост - только на предыдущий. Это делает возможным быстрое добавление и удаление элементов как в начале, так и в конце списка.

* **begin** – указатель на первый элемент списка;
* **end** – указатель на последний элемент списка;

Общий вид двусвязного списка:



Также будем использовать счетчик (count) для подсчета количества элементов в списке.

**Создание двусвязного списка**

При создании списка перед добавлением новых элементов важно начать с формирования пустого списка, который будет готов принимать и хранить данные. Этот процесс обычно включает в себя следующие этапы:

1. Объявление указателей на начало и конец списка (**begin**, **end**).

Item<T>\* begin;

Item<T>\* end;

2. Установление нулевых значений указателям.

begin = nullptr;

end = nullptr;

3. Установка счетчика длины списка (**count**) в нулевое значение (для нашего удобства).

int count;

count = 0;

Такой подход к созданию списка обеспечивает его корректное начальное состояние и готовность к приему новых элементов.

**Добавление элемента**

Добавление элемента в двусвязный список может происходить в различных сценариях, в зависимости от текущего состояния списка. Рассмотрим два основных случая: добавление элемента в пустой список и добавление элемента в список, содержащий элементы.

**Добавление элемента в пустой список**

При добавлении элемента в пустой список необходимо учитывать, что в списке отсутствуют другие элементы. Этот случай требует особого внимания, так как в этом случае новый элемент становится как первым, так и последним элементом списка.

Алгоритм добавления будет выглядеть следующим образом:

* создать новый элемент и заполнить его данными;
* установить начало и конец списка так, чтобы они указывали на новый элемент;
* увеличить общее количество элементов в списке.

Item<T>\* elem = new Item<T>;  // Создание нового элемента списка и выделение памяти для него

element->next = nullptr;      // Установка указателя на следующий элемент в nullptr, так как элемент будет последним в списке

element->prev = nullptr;      // Установка указателя на предыдущий элемент в nullptr, так как элемент будет первым в списке

element->data = \_data;        // Присвоение значению элемента переданного значения \_data

begin = end = element;  // Установка указателей begin и end на новый элемент списка element, так как он является единственным элементом в списке

**Добавление элемента в список, содержащий элементы**

При добавлении элемента в список, который уже содержит другие элементы, новый элемент добавляется в конец списка. Для этого необходимо обновить указатели предыдущего последнего элемента списка и указателя на хвост.

Алгоритм добавления будет выглядеть следующим образом:

* создание нового элемента. Заполнение полей **data** и **next** элемента;
* настройка указателя **prev** элемента таким образом, что он указывает на текущую позицию указателя конца списка;
* настройка указателя **next** последнего элемента таким образом, чтобы он указывал элемент;
* смещение указателя конца списка на одну позицию, определенную указателем **element**.

Item<T>\* elem = new Item<T>;  // Создание нового элемента списка и выделение памяти для него

element->next = nullptr;                // Установка указателя на следующий элемент в nullptr, так как элемент будет добавлен в конец списка

element->data = \_data;                  // Присвоение значению элемента переданного значения \_data

element->prev = end;                    // Установка указателя на предыдущий элемент списка на текущий хвост списка

end->next = elem;                    // Обновление указателя next текущего хвоста списка, чтобы он указывал на новый элемент

end = element;                          // Обновление указателя на хвост списка, чтобы он указывал на новый элемент

**Преимущества и недостатки двусвязного списка**

Преимущества двусвязных списков:

1. Быстрое добавление и удаление элементов: В отличие от массивов, где добавление или удаление элементов может потребовать переноса всего массива, в двусвязных списках добавление и удаление элементов требует только изменения указателей, что делает эти операции более эффективными.

2. Гибкость при обходе элементов: Двусвязные списки обеспечивают возможность обхода элементов как в прямом, так и в обратном направлении благодаря наличию указателей на предыдущий и следующий элементы.

3. Эффективное использование памяти: Память в двусвязных списках выделяется динамически по мере необходимости, что позволяет эффективно использовать память и избежать предварительного выделения фиксированного размера, как в случае с массивами.

4. Устойчивость к фрагментации памяти: Поскольку память выделяется по мере необходимости, а элементы списка могут быть удалены и добавлены в любом месте, двусвязные списки более устойчивы к фрагментации памяти, чем некоторые другие структуры данных.

Недостатки двусвязных списков:

1. Дополнительное использование памяти: Каждый узел двусвязного списка требует дополнительной памяти для хранения указателей на предыдущий и следующий элементы. Это может привести к некоторому избыточному использованию памяти, особенно для больших списков.

2. Более сложная реализация: Работа с указателями на предыдущий и следующий элементы требует более сложной реализации и может потребовать более тщательного контроля за указателями, чтобы избежать проблем с утечкой памяти или некорректным доступом к памяти.

3. Более медленный доступ к элементам по индексу: Доступ к элементам в двусвязном списке по индексу может быть менее эффективным, чем в массивах, поскольку требуется обходить список от начала или конца до нужного элемента.

**Заключение**

В заключении, двусвязные списки представляют собой важную структуру данных, которая обладает как преимуществами, так и недостатками. Они широко используются в программировании благодаря своей гибкости и эффективности в операциях добавления и удаления элементов. Однако, при выборе использования двусвязного списка необходимо учитывать его особенности и осознанно оценивать потребность в нем.

Преимущества двусвязных списков включают быстрые операции добавления и удаления элементов, гибкость при обходе элементов, эффективное использование памяти и устойчивость к фрагментации памяти. Однако, они также имеют недостатки, такие как дополнительное использование памяти, более сложная реализация и более медленный доступ к элементам по индексу.

В целом, двусвязные списки представляют собой мощный инструмент для реализации различных задач, таких как управление данными, реализация алгоритмов и создание структур данных. Понимание и учет их преимуществ и недостатков поможет разработчикам эффективно использовать их в своих проектах, обеспечивая эффективность и гибкость в обработке данных.

**Источники**

1. Шаблон класса. URL: https://metanit.com/cpp/tutorial/9.1.php
2. Связный список и двусвязный список. URL: https://javarush.com/quests/lectures/questharvardcs50.level05.lecture05
3. Шаблонные классы. URL: https://education.yandex.ru/handbook/cpp/article/template-classes
4. Реализация двусвязого списка на C++. URL: https://pvs-studio.ru/ru/blog/terms/6683/
5. Двусвязный линейный список. URL: https://prog-cpp.ru/data-dls/
6. Двусвязный список. URL: https://wiki.dieg.info/dvusvjaznyj\_spisok
7. Линейный двухсвязный (двунаправленный) список. Общие сведения. URL: https://www.bestprog.net/ru/2022/02/16/c-linear-doubly-linked-bidirectional-list-general-concepts-ru/#contents