1 Системы методов линейно организованных Абстрактных Типов Данных

1.1 Введение

В данном документе приведены внешние описания (***интерфейсная часть***) АТД типа «Стек» / «Очередь» / «Дек» / «Последовательность» / «Л-списки», реализация функционирования которых осуществлена в библиотеке « ***библиотекаАТД*** ».

В первых разделах документа (***1.2*** – ***1.7***) приводятся сведения о совокупности методов и свойств (***интерфейсе***) параметризированных классов:

- Стек – ***Стек<T>***;
- Очередь – ***Очередь<T>***;
- Дек – ***Дек<T>***;
- Последовательность – ***Последовательность<T>***;
- Л1-список – ***Л1-список<T>*** и
- Л2-список – ***Л2-список<T>***,

где ***T*** - тип элементов соответствующей структуры.

Заметим, что ***T*** может быть любым типом, как определенным в ***C#***, т.е. стандартным, так и описанным пользователем.

Все множество средств разделим на отдельные группы, реализующих специфические действия.

1.2 Группы методов и свойств класса *Стек <T>*

1.2.1 Специальные методы

1. **public** Стек(params T[] нагрузка)

   Конструктор создания изначально заполненного объекта класса *«Стек»*.

   Начальные значения : множество элементов, значения которых изначально заносятся в структуру (допустимо пустое множество - пустой стек)

   Процесс : инициализация полей данных объекта.

2. **public void** пустойСтек()

   Метод удаления всех элементов структуры и инициализация полей объекта.

   Вход : нет;

   Предусловие : нет;

   Процесс : удаление элементов стека и инициализация полей;

   Постусловие : нет;

   Выход : объект инициализирован.

3. **public void** показать()

   Вспомогательный метод контрольного просмотра (с выводом на консоль) элементов объекта *«Стек»* без его «разрушения».

   Вход : нет;

   Предусловие : нет;

   Процесс : вывод на консоль значений элементов объекта от начала структуры;

Постусловие : нет;

Выход : состояние объекта не изменилось.

1.2.2 Методы обработки элементов структуры

1. **public void** вСтек(**T** нагрузка)

   Метод добавления в конец структуры «Стек» нового элемента

   Вход : нет;

   Предусловие : нет;

   Процесс : добавление в конец структуры элемента со значением *нагрузка*;

   Постусловие : сменилось значение «*вершинного*» элемента;

   Выход : структура увеличила свою мощность на единицу.

2. **public void** удалитьНачало()

   Метод удаляет «*вершинный*» элемент структуры.

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : удаление «*вершинного*» элемента из конца структуры;

   Постусловие : сменилось значение «*вершинного*» элемента;

   Выход : структура уменьшила свою мощность на единицу.

1.2.3 Свойства обработки элементов структуры

1. **public bool** стекПуст

   Свойство-определитель пустоты структуры «*Стек*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуст ли стек ?*»;

   Постусловие : нет;

   Выход : **true** – если стек пуст и **false** – если стек не пуст.

2. **public bool** стекНеПуст

   Свойство-определитель непустоты структуры «*Стек*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуст ли стек ?*»;

   Постусловие : нет;

   Выход : **true** – если стек не пуст и **false** – если стек пуст.

3. **public T** началоСтека

   Свойство-определитель значения «*вершинного*» элемента структуры

   Вход : нет;

Предусловие : структура *не пуста*;

Процесс : получение значения «*вершинного*» элемента структуры;

Постусловие : нет;

Выход : значение «*вершинного*» элемента структуры.

4. **public T** изСтека

Свойство извлечения «*вершинного*» элемента структуры

Вход : нет;

Предусловие : структура *не пуста*;

Процесс : получение значения «*вершинного*» элемента структуры;

Постусловие : структура уменьшила свою мощность на единицу;

Выход : значение «*вершинного*» элемента структуры.

1.3 Группы методов и свойств класса *Очередь <T>*

1.3.1 Специальные методы

1. **public** Очередь(params T[] нагрузка)

   Конструктор создания изначально заполненного объекта класса *«Очередь»*.

   Начальные значения : множество элементов, значения которых изначально заносятся в структуру (допустимо пустое множество - пустая очередь)

   Процесс : инициализация полей данных объекта.

2. **public void** пустаяОчередь()

   Метод удаления всех элементов структуры и инициализация полей объекта.

   Вход : нет;

   Предусловие : нет;

   Процесс : удаление элементов очереди и инициализация полей;

   Постусловие : нет;

   Выход : объект инициализирован.

3. **public void** показать()

   Вспомогательный метод контрольного просмотра (с выводом на консоль) элементов объекта *«Очередь»* без его «разрушения».

   Вход : нет;

   Предусловие : нет;

   Процесс : вывод на консоль значений элементов объекта от начала структуры;

   Постусловие : нет;

   Выход : состояние объекта не изменилось.

1.3.2 Методы обработки элементов структуры

1. **public void** вОчередь(**T** нагрузка)

   Метод добавления в конец структуры «Очередь» нового элемента

   Вход : нет;

   Предусловие : нет;

   Процесс : добавление в конец структуры элемента со значением *нагрузка*;

   Постусловие : сменилось значение «*конечного*» элемента;

   Выход : структура увеличина свою мощность на единицу.

2. **public void** удалитьНачало()

   Метод удаляет «*начальный*» элемент структуры.

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : удаление «*начального*» элемента из начала структуры;

   Постусловие : сменилось значение «*начального*» элемента;

   Выход : структура уменьшила свою мощность на единицу.

1.3.3 Свойства обработки элементов структуры

1. **public bool** очередьПуста

   Свойство-определитель пустоты структуры «*Очередь*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуста ли очередь ?*»;

   Постусловие : нет;

   Выход : **true** – если очередь пуста и **false** – если очередь не пуста.

2. **public bool** очередьНеПуста

   Свойство-определитель непустоты структуры «*Очередь*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуста ли очередь ?*»;

   Постусловие : нет;

   Выход : **true** – если очередь не пуста и **false** – если очередь пуста.

3. **public T** началоОчереди

   Свойство-определитель значения «*начального*» элемента структуры

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : получение значения «*начального*» элемента структуры;

   Постусловие : нет;

   Выход : значение «*начального*» элемента структуры.

4. **public T** изОчереди

   Свойство извлечения «*начального*» элемента структуры

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : получение значения «*начального*» элемента структуры;

   Постусловие : структура уменьшила свою мощность на единицу;

   Выход : значение «*начального*» элемента структуры.

1.4 Группы методов и свойств класса *Дек <T>*

1.4.1 Специальные методы

1. **public** Дек(params T[] нагрузка)

   Конструктор создания изначально заполненного объекта класса *«Дек»*.

   Начальные значения : множество элементов, значения которых изначально заносятся в структуру (допустимо пустое множество - пустой дек)

   Процесс : инициализация полей данных объекта.

2. **public void** пустойДек()

   Метод удаления всех элементов структуры и инициализацией полей объекта.

   Вход : нет;

   Предусловие : нет;

   Процесс : удаление элементов дека и инициализация полей;

   Постусловие : нет;

   Выход : объект инициализирован.

3. **public void** показать()

   Вспомогательный метод контрольного просмотра (с выводом на консоль) элементов объекта *«Дек»* без его «разрушения».

   Вход : нет;

   Предусловие : нет;

   Процесс : вывод на консоль значений элементов объекта от начала структуры;

   Постусловие : нет;

   Выход : состояние объекта не изменилось.

1.4.2 Методы обработки элементов структуры

1. **public void** вНачалоДека(**T** нагрузка)

   Метод добавления в начало структуры «Дек» нового элемента

   Вход : нет;

   Предусловие : нет;

   Процесс : добавление в начало структуры элемента со значением *нагрузка*;

   Постусловие : сменилось значение «*начального*» элемента;

   Выход : структура увеличила свою мощность на единицу.

2. **public void** вКонецДека(**T** нагрузка)

   Метод добавления в конец структуры «Дек» нового элемента

   Вход : нет;

   Предусловие : нет;

   Процесс : добавление в конец структуры элемента со значением *нагрузка*;

   Постусловие : сменилось значение «*конечного*» элемента;

   Выход : структура увеличила свою мощность на единицу.

3. **public void** удалитьНачало()

   Метод удаляет «*начальный*» элемент структуры.

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : удаление «*начального*» элемента структуры;

   Постусловие : сменилось значение «*начального*» элемента;

   Выход : структура уменьшила свою мощность на единицу.

4. **public void** удалитьКонец()

   Метод удаляет «*конечный*» элемент структуры.

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : удаление «*конечного*» элемента структуры;

   Постусловие : сменилось значение «*конечного*» элемента;

   Выход : структура уменьшила свою мощность на единицу.

1.4.3 Свойства обработки элементов структуры

1. **public bool** декПуст

   Свойство-определитель пустоты структуры «*Дек*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуст ли дек ?*»;

   Постусловие : нет;

   Выход : **true** – если дек пуст и **false** – если дек не пуст.

2. **public bool** декНеПуст

   Свойство-определитель непустоты структуры «*Дек*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуст ли дек ?*»;

   Постусловие : нет;

   Выход : **true** – если дек не пуст и **false** – если дек пуст.

3. **public T** началоДека

   Свойство-определитель значения «*начального*» элемента структуры

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : получение значения «*начального*» элемента структуры;

   Постусловие : нет;

   Выход : значение «*начального*» элемента структуры.

4. **public T** конецДека

   Свойство-определитель значения «*конечного*» элемента структуры

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : получение значения «*конечного*» элемента структуры;

   Постусловие : нет;

   Выход : значение «*конечного*» элемента структуры.

5. **public T** изНачалаДека

   Свойство извлечения «*начальноого*» элемента структуры

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : получение значения «*начальноого*» элемента структуры;

   Постусловие : структура уменьшила свою мощность на единицу;

   Выход : значение «*начальноого*» элемента структуры.

6. **public T** изКонцаДека

   Свойство извлечения «*конечного*» элемента структуры

   Вход : нет;

   Предусловие : структура *не пуста*;

   Процесс : получение значения «*конечного*» элемента структуры;

   Постусловие : структура уменьшила свою мощность на единицу;

   Выход : значение «*конечного*» элемента структуры.

1.5 Группы методов и свойств класса *Последовательность<T>*

1.5.1 Специальные методы

1. **public** Последовательность(params T[] нагрузка)

   Конструктор создания изначально заполненного объекта класса *«Последовательность»*.

   Начальные значения : множество элементов, значения которых изначально заносятся в структуру (допустимо пустое множество - пустая последовательность)

   Процесс : инициализация полей данных объекта.

2. **public void** пустаяПоследовательность()

   Метод удаления всех элементов структуры и инициализация полей объекта.

   Вход : нет;

   Предусловие : нет;

   Процесс : удаление элементов последовательности и инициализация полей;

   Постусловие : нет;

   Выход : объект инициализирован.

3. **public void** показать()

   Вспомогательный метод контрольного просмотра (с выводом на консоль) элементов объекта *«Последовательность»* без его «разрушения».

   Вход : нет;

   Предусловие : нет;

   Процесс : вывод на консоль значений элементов объекта от начала структуры;

   Постусловие : нет;

   Выход : состояние объекта не изменилось.

1.5.2 Методы обработки элементов структуры

1. **public void** вПоследовательность(**T** нагрузка)

   Метод добавления в конец структуры « Последовательность» нового элемента

   Вход : нет;

   Предусловие : нет;

   Процесс : добавление в конец структуры элемента со значением *нагрузка*;

   Постусловие : структура пополнилась «*с конца*» новым элементом;

   Выход : структура увеличила свою мощность на единицу.

2. **public void** вНачалоПоследовательности()

   Метод переустанавливает указатель на «*начальный*» элемент структуры.

   Вход : нет;

   Предусловие : нет;

   Процесс : переустановка указателя на «*начальный*» элемент структуры;

   Постусловие : сменилось значение указателя структуры;

   Выход : нет.

3. **public void** пропуститьОчередной()

   Метод переустанавливает указатель на «*следующий*» элемент структуры.

   Вход : нет;

   Предусловие : указатель не в конце последовательности;

   Процесс : переустановка указателя на «*следующий*» элемент структуры;

   Постусловие : сменилось значение указателя структуры;

   Выход : нет.

1.5.3 Свойства обработки элементов структуры

1. **public bool** последовательностьПуста

   Свойство-определитель пустоты структуры «*Последовательность*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуста ли последовательность ?*»;

   Постусловие : нет;

   Выход : **true** – если последовательность пуста и **false** – если последовательность не пуста.

2. **public bool** последовательностьНеПуста

   Свойство-определитель непустоты структуры «*Последовательность*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуста ли последовательность ?*»;

   Постусловие : нет;

   Выход : **true** – если последовательность не пуста и **false** – если последовательность пуста.

3. **public bool** естьНепрочитанныеЭлементы

   Свойство-определитель наличия элементов «*за указателем*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*есть ли элементы за указателем последовательности ?*»;

   Постусловие : нет;

   Выход : **true** – если указатель не в конце последовательности и **false** – если указатель в конце последовательности.

4. **public T** очереднойЭлемент

   Свойство-определитель значения «*очередного*» элемента структуры

   Вход : нет;

   Предусловие : указатель *не в конце последовательности*;

   Процесс : получение значения «*очередного*» элемента структуры;

   Постусловие : нет;

   Выход : значение «*очередного*» элемента структуры.

5. **public T** прочестьОчередной

Свойство извлечения значения «*очередного*» элемента структуры, «*указатель вперед*»

Вход : нет;

Предусловие : структура *не пуста*;

Процесс : получение значения «*очередного*» элемента структуры со сменой положения указателя;

Постусловие : указатель сместился «за очередной» элемент структуры ;

Выход : значение «*очередного*» элемента структуры.

1.6 Группы методов и свойств класса *Л1_ список<T>*

1.6.1 Специальные методы

1. **public** Л1_список(params T[] нагрузка)

   Конструктор создания изначально заполненного объекта класса *«Л1_ список»*.

   Начальные значения : множество элементов, значения которых изначально заносятся в структуру (допустимо пустое множество - пустой список)

   Процесс : инициализация полей данных объекта.

2. **public void** пустойСписок()

   Метод удаления всех элементов структуры и инициализация полей объекта.

   Вход : нет;

   Предусловие : нет;

   Процесс : удаление элементов списка и инициализация полей;

   Постусловие : нет;

   Выход : объект инициализирован.

3. **public void** показать()

   Вспомогательный метод контрольного просмотра (с выводом на консоль) элементов объекта *«Л1_ списка»* без его «разрушения».

   Вход : нет;

   Предусловие : нет;

   Процесс : вывод на консоль значений элементов объекта от начала структуры;

   Постусловие : нет;

   Выход : состояние объекта не изменилось.

1.6.2 Методы обработки элементов структуры

1. **public void** добавитьЗаУказателем(**T** нагрузка)

   Метод добавления нового элемента в « Л1_список» *за указателем*

   Вход : нет;

   Предусловие : нет;

   Процесс : добавление элемента со значением *нагрузка* в структуру «*за*» текущим положением указателя;

   Постусловие : структура пополнилась «*за*» указателем новым элементом;

   Выход : структура увеличила свою мощность на единицу.

2. **public void** удалитьЗаУказателем()

   Метод удаляет элемент структуры, расположенный «*за*» текущим положением указателя.

   Вход : нет;

   Предусловие : указатель не в конце списка;

   Процесс : уделение элемента структуры, расположенного «*за*» текущим положением указателя;

Постусловие : сменилось значение указателя структуры, мощность структуры уменьшилась на единицу;

Выход : нет.

3. **public void** вНачалоСписка()

   Метод устанавливает указатель перед «*начальным*» элементом структуры.

   Вход : нет;

   Предусловие : нет;

   Процесс : установка указателя перед «*начальным*» элементом структуры;

   Постусловие : сменилось значение указателя структуры;

   Выход : нет.

4. **public void** указательВперед()

   Метод переустанавливает указатель «*на шаг вперед*» к концу списка.

   Вход : нет;

   Предусловие : указатель не в конце списка;

   Процесс : переустановка указателя «*на шаг вперед*» к концу списка;

   Постусловие : сменилось значение указателя структуры;

   Выход : нет.

1.6.3 Свойства обработки элементов структуры

1. **public bool** списокПуст

   Свойство-определитель пустоты структуры «*Л1_список*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуст ли Л1_список ?*»;

   Постусловие : нет;

   Выход : **true** – если список пуст и **false** – если список не пуст.

2. **public bool** списокНеПуст

   Свойство-определитель непустоты структуры «*Л1_список*»

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*не пуст ли список ?*»;

   Постусловие : нет;

   Выход : **true** – если список не пуст и **false** – если список пуст.

3. **public bool** указательВКонце

   Свойство-определитель положения указателя

   Вход : нет;

   Предусловие : нет;

Процесс : проверка «*есть ли элементы за текущим положением указателя списка ?*»;

Постусловие : нет;

Выход : **true** – если указатель в конце списка и **false** – если указатель не в конце списка.

4. **public T** элементЗаУказателем

   Свойство-определитель значения элемента структуры «*за*» текущим положением указателя;

   Вход : нет;

   Предусловие : указатель *не в конце списка*;

   Процесс : получение значения элемента структуры «*за*» текущим положением указателя;

   Постусловие : нет;

   Выход : значение элемента структуры «*за*» текущим положением указателя.

5. **public T** взятьЗаУказателем

   Свойство извлечения значения элемента структуры, расположенного «*за*» текущим положением указателя.

   Вход : нет;

   Предусловие : указатель *не в конце списка*;

   Процесс : получение значения элемента структуры, расположенного «*за*» текущим положением указателя, со сменой положения указателя «на шаг вперед» к концу списка;

   Постусловие : указатель сместился на элемент структуры, расположенный «за» текущим положением указателя;

   Выход : значение элемента структуры, расположенного «*за*» текущим положением указателя.

1.7 Группы методов и свойств класса *Л2_ список<T>*

1.7.1 Специальные методы

1. **public** Л2_список(params T[] нагрузка)

   Конструктор создания изначально заполненного объекта класса *«Л2_ список»*.

   Начальные значения : множество элементов, значения которых изначально заносятся в структуру (допустимо пустое множество - пустой список)

   Процесс : инициализация полей данных объекта.

2. **public void** пустойСписок()

   Метод удаления всех элементов структуры и инициализация полей объекта.

   Вход : нет;

   Предусловие : нет;

   Процесс : удаление элементов списка и инициализация полей;

   Постусловие : нет;

   Выход : объект инициализирован.

3. **public void** показать()

   Вспомогательный метод контрольного просмотра (с выводом на консоль) элементов объекта *«Л2_ списка»* без его «разрушения».

   Вход : нет;

   Предусловие : нет;

   Процесс : вывод на консоль значений элементов объекта от начала структуры;

   Постусловие : нет;

   Выход : состояние объекта не изменилось.

1.7.2 Методы обработки элементов структуры

1. **public void** добавитьЗаУказателем(**T** нагрузка)

   Метод добавления нового элемента в « Л2_список» *за указателем*

   Вход : нет;

   Предусловие : нет;

   Процесс : добавление элемента со значением *нагрузка* в структуру «*за*» текущим положением указателя;

   Постусловие : структура пополнилась «*за*» указателем новым элементом;

   Выход : структура увеличила свою мощность на единицу.

2. **public void** добавитьПередУказателем(**T** нагрузка)

   Метод добавления нового элемента в « Л2_список» *перед указателем*

   Вход : нет;

   Предусловие : нет;

   Процесс : добавление элемента со значением *нагрузка* в структуру «*перед*» текущим положением указателя;

Постусловие : структура пополнилась «*перед*» указателем новым элементом;

Выход : структура увеличила свою мощность на единицу.

3. **public void** удалитьЗаУказателем()

   Метод удаляет элемент структуры, расположенный «*за*» текущим положением указателя.

   Вход : нет;

   Предусловие : указатель не в конце списка;

   Процесс : уделение элемента структуры, расположенного «*за*» текущим положением указателя;

   Постусловие : сменилось значение указателя структуры, мощность структуры уменьшилась на единицу;

   Выход : нет.

4. **public void** удалитьПередУказателем()

   Метод удаляет элемент структуры, расположенный «*перед*» текущим положением указателя.

   Вход : нет;

   Предусловие : указатель не в начале списка;

   Процесс : уделение элемента структуры, расположенного «*перед*» текущим положением указателя;

   Постусловие : сменилось значение указателя структуры, мощность структуры уменьшилась на единицу;

   Выход : нет.

5. **public void** вНачалоСписка()

   Метод устанавливает указатель перед «*начальным*» элементом структуры.

   Вход : нет;

   Предусловие : нет;

   Процесс : установка указателя перед «*начальным*» элементом структуры;

   Постусловие : сменилось значение указателя структуры;

   Выход : нет.

6. **public void** вКонецСписка()

   Метод устанавливает указатель за «*последним*» элементом структуры.

   Вход : нет;

   Предусловие : нет;

   Процесс : установка указателя за «*последним*» элементом структуры;

   Постусловие : сменилось значение указателя структуры;

   Выход : нет.

7. **public void** указательВперед()

   Метод переустанавливает указатель «*на шаг вперед*» к концу структуры.

   Вход : нет;

Предусловие : указатель не в конце списка;

Процесс : переустановка указателя «*на шаг вперед*» к концу структуры;

Постусловие : сменилось значение указателя структуры;

Выход : нет.

8. **public void** указательНазад()

Метод переустанавливает указатель «*на шаг назад*» началу структуры.

Вход : нет;

Предусловие : указатель не в начале списка;

Процесс : переустановка указателя на «*на шаг назад*» началу структуры;

Постусловие : сменилось значение указателя структуры;

Выход : нет.

1.7.3 Свойства обработки элементов структуры

1. **public bool** списокПуст

Свойство-определитель пустоты структуры «*Л2_ список*»

Вход : нет;

Предусловие : нет;

Процесс : проверка «*не пуст ли Л2_ список ?*»;

Постусловие : нет;

Выход : **true** – если список пуст и **false** – если список не пуст.

2. **public bool** списокНеПуст

Свойство-определитель непустоты структуры «*Л2_ список*»

Вход : нет;

Предусловие : нет;

Процесс : проверка «*не пуст ли список ?*»;

Постусловие : нет;

Выход : **true** – если список не пуст и **false** – если список пуст.

3. **public bool** указательВКонце

Свойство-определитель положения указателя

Вход : нет;

Предусловие : нет;

Процесс : проверка «*есть ли элементы за текущим положением указателя списка ?*»;

Постусловие : нет;

Выход : **true** – если указатель в конце списка и **false** – если указатель не в конце списка.

4. **public bool** указательВНачале

   Свойство-определитель положения указателя

   Вход : нет;

   Предусловие : нет;

   Процесс : проверка «*есть ли элементы перед текущим положением указателя списка ?*»;

   Постусловие : нет;

   Выход : **true** – если указатель в начале списка и **false** – если указатель не в начале списка.

5. **public T** элементЗаУказателем

   Свойство-определитель значения элемента структуры «*за*» текущим положением указателя;

   Вход : нет;

   Предусловие : указатель *не в конце списка*;

   Процесс : получение значения элемента структуры «*за*» текущим положением указателя;

   Постусловие : нет;

   Выход : значение элемента структуры «*за*» текущим положением указателя.

6. **public T** элементПередУказателем

   Свойство-определитель значения элемента структуры «*перед*» текущим положением указателя;

   Вход : нет;

   Предусловие : указатель *не в начале списка*;

   Процесс : получение значения элемента структуры «*перед*» текущим положением указателя;

   Постусловие : нет;

   Выход : значение элемента структуры «*перед*» текущим положением указателя.

7. **public T** взятьЗаУказателем

   Свойство извлечения значения элемента структуры, расположенного «*за*» текущим положением указателя.

   Вход : нет;

   Предусловие : указатель *не в конце списка*;

   Процесс : получение значения элемента структуры, расположенного «*за*» текущим положением указателя, со сменой положения указателя;

   Постусловие : указатель сместился на элемент структуры, расположенный «за» текущим положением указателя;

   Выход : значение элемента структуры, расположенного «*за*» текущим положением указателя.

8. **public T** взятьПередУказателем

   Свойство извлечения значения элемента структуры, расположенного «*перед*» текущим положением указателя.

   Вход : нет;

   Предусловие : указатель *не в начале списка*;

Процесс : получение значения элемента структуры, расположенного «*перед*» текущим положением указателя, со сменой положения указателя;

Постусловие : указатель сместился на элемент структуры, расположенный «перед» текущим положением указателя;

Выход : значение элемента структуры, расположенного «*перед*» текущим положением указателя.

1.8 Особенности работы с библиотекой классов «библиотекаАТД»

1. В пункте меню «**Проект**» выбрать пункт «**Добавить ссылку**», а по открытии формы – подпункт «**Обзор**». Далее, найти файл «*библиотекаАТД.dll*»[1]

2. Раздел подключения библиотек классов дополнить строкой «**using** библиотекаАТД;»

3. В блоке класса «**class** *Program*» вставить строку объявления и инициализации объекта требуемого класса (Стек / Очередь / Дек / Последовательность / Л1_список / Л2_список), с конкретизацией типа **T**, например,

   (a) static Стек<int> стек = new Стек<int>();

   (b) static Очередь<double> очередь = new Очередь<double>();

   (c) static Дек<byte> дек = new Дек<byte>();

   (d) static Последовательность<char> = new Последовательность<char>();

   (e) static Л1_список<char> = new Л1_список<char>('Л','1','_','с','п','и','с','о','к');

   (f) static Л2_список<string> = new Л2_список<string>("АТД ,"Стек "Очередь "Дек ".");

4. Все классы самодокументированы, поэтому при выборе соответствующих методов рекомендуется следить за «подсказками» среды программирования.

5. Пример оформления кода программы следующий[2]

```
using System;
using System.IO;
using библиотекаАТД;

namespace ADT_Stack_Задача_01
{
    class Program
    {
        /*
         *    Текстовый файл Inlet.in содержит целочисленные значения. Заполнить элементами названного файла
         *    стек целочисленных элементов.
         *    Верно ли, что все его элементы различны? Результат решения задачи вывести в текстовый файл Outlet.out.
         *
         *     Замечание.
         *        В задаче разрешается использовать один дополнительный стек элементов целочисленного типа.
         *
         *     Спецификация ввода (файл Inlet.in):
         *        Целочисленные значения (по одному в строке)
         *     Спецификация вывода (файл Outlet.out):
         *         Yes или No
         */

        static Стек<int> стек, вспомогательныйСтек;
        static StreamReader файлВв = new StreamReader("Inlet.in");
        static StreamWriter файлВы = new StreamWriter("Outlet.out");

        static void Main(string[] args)
        {
            стек = new Стек<int>();
            вводДанных();
            //стек.показать();
            файлВы.Write(анализСтека());
            файлВы.Close();
        }
        static string анализСтека()
```

[1]Адрес файла следует узнать у преподавателя.

[2]Приведено решение задачи № 1 из списка задач лабораторных работ по теме «АТД Стек».

```
        {
            вспомогательныйСтек = new Стек<int>();
            bool ключУникален = true;
            string ответ = "No";
            int ключПоиска = стек.изСтека;
            while (стек.стекНеПуст && ключУникален)
            {
                сравнениеКлюча_с_ОстальнымиЭлементами(ключПоиска, ref ключУникален);
                if (ключУникален)
                {
                    перекачкаИзВспомогательногоСтека();
                    ключПоиска = стек.изСтека;
                }
                else
                    ответ = "Yes";
            }
            return ответ;
        }
        static void сравнениеКлюча_с_ОстальнымиЭлементами(int ключПоиска, ref bool ключУникален)
        {
            while (стек.стекНеПуст && ключУникален)
            {
                int элемент = стек.изСтека;
                ключУникален = элемент != ключПоиска;
                if (ключУникален)
                    вспомогательныйСтек.вСтек(элемент);
            }
        }
        static void перекачкаИзВспомогательногоСтека()
        {
            while (вспомогательныйСтек.стекНеПуст)
                стек.вСтек(вспомогательныйСтек.изСтека);
        }
        static void вводДанных()
        {
            while (!файлВв.EndOfStream)
                стек.вСтек(Convert.ToInt16(файлВв.ReadLine()));
        }
    }
}
```