Постановка задачі

Списки та функціональне програмування. Мови функціонального програмування (Lisp, ML, Haskell та ін.) містять єдину структура даних – список і єдину алгоритмічну структуру – функцію, яка також має вигляд списку. Необхідно скласти програму, яка забезпечує роботу користувача зі списками у стилі функціонального програмування.

Користувач вводить:

a) (= x (1 2 3)) - програма виконує наступні дії:у список *x* заносяться значення (1,2,3).

b) (cdr x) – в результаті на екран виводиться хвіст списку

c) (= y (append x (-2 -1)) – в результаті список *y* буде містити значення (1 2 3 -2 -1)

Використання псевдозмінних для списку аналогічно використанню псевдозмінних для стека.

Розробити об'єктно-орієнтовану бібліотеку для роботи зі структурою даних список. Властивості та методи для класів розробити у відповідності з відомими визначеннями. Скласти тести для перевірки працездатності бібліотеки. Скласти програму, в якій реалізовані вищєзазначені операції роботи зі списками

У класі списку повинні бути реалізовані наступні можливості:

1. Реалізація методів ініціалізації (конструктор по замовчуванню та конструктор з параметрами), копіювання (конструктор копіювання), індексації (перевантаження []), присваивания (перевантаження =), візуалізації, збереження (на диск) та відновлення, діалогового керування, "розумного доступу" (перевантаження ->), а також псевдо змінних (забезпечення можливості виду: f(x)=const).
2. Перевантаження (спільне використання) потокового введення/виведення. (введення з файл, виведення в файл)
3. Створення та використання файла бібліотеки (*.dylib).
4. Повторне використання класів без їх перекомпіляції (ReUse).
5. Застосування структури список для розв'язання задачі.

Опис розв'язку

Для розв'язку задачі було створено структуру *ListNode* - вузол списку, що містить значення елементу(ціле число) та покажчик на наступний вузол та клас *IntList*, в якому і були реалізовані всі необхідні операції роботи зі списками:

1. IntList(); - конструктор за замовчуванням
2. IntList(int); - конструктор з параметром, значення параметру - значення голови списку
3. IntList(std::string); - конструктор з параметром, значення параметру - строка вигляду (1 2 3), задає значення списку
4. IntList(IntList&); - конструктор копіювання
5. ~IntList(); - деструктор
6. virtual void push(int); - додає новий елемент у хвіст списку
7. virtual int pop(); - виштовхує останній елемент з хвосту списку
8. virtual bool isEmpty(); - перевіряє чи пустий список
9. virtual std::string toString(); - представлення списку у вигляді строки (1 2 3 4)
10. virtual std::vector<int> toVector(); - список у вигляді std::vector<int>
11. virtual std::string printTail(); - хвіст списку у вигляді строки
12. int& operator[](int); - перевантаження оператору індексування
13. IntList& operator=(const IntList&); - перевантаження оператору присваювання

14. IntList& operator->(); - перевантаження оператору ->
15. IntList& operator()(const int s, const int e); - перевантаження оператору (). Повертає новий список, що містить елементи даного з індексу s по e
16. std::ostream& operator <<(std::ostream&, IntList&); - перевантаження потокового виведення
17. std::istream& operator >>(std::istream&, IntList&); - перевантаження потокового введення

Для повторного використання класу він був виділений у окрему бібліотку *libintlist.dylib*, яку можливо динамічно зв'язувати з іншими програмами та використовувати в них клас *IntList*.

Для тестування всіх можливостей класу перед запуском програми на екран виводяться результати тестів. Для тесту потокового введення необхідно втручання користувача(введення списку с клавіатури):

```
running some tests...
default constructor:
list.toString() method: ()
list.push:
push(20)
push(30)
push(40)
(20 30 40)
indexing:
list[2] => 40
list[2] <= 3 : (20 30 3)
list.pop():
pop():3
pop():30
pop():20
constructor with params (1 25 8):
stream output: list2 = (1 25 8)
copy-constructor: IntList list3(list2) :(1 25 8)
pseudo-variables: list3(0,2):(1 25)
stream input:(5 6 7)
list = (5 6 7)
writing list to file "list.txt" ok
reading list from file "list.txt": (5 6 7)
```

Після проведення демонстрації можливостей класу, програма переходить до діалового режиму керування списками.

Вихідний текст програми розв'язку задачі

див. Додаток 1

Опис інтерфейсу (керівництво користувача)

Програма дозволяє оперувати зі списками у стилі функціонального програмування в діалоговому режимі: користувач вводить команду з клавіатури, після чого вона обробляється і на екран виводиться її результат(у випадку створення нового списку - його строкове

представлення, у випадку виведення хвосту списку - хвіст у вигляді строки). Команди роботи зі списками визначені у поставновці задачі.

Опис тестових прикладів

```
>> (= x (1 2 3))
x = (1 2 3)
>> (= y (append x (4 5 6)))
y = (1 2 3 4 5 6)
>> (cdr y)
(2 3 4 5 6)
```

результати роботи повністю відповідають постановці задачі.