LISTA

O strukturze danych:

Lista jest to liniowo uporządkowany zbiór elementów, z których dowolny element można usunąć oraz dodać w dowolnym miejscu. Pierwszy i ostatni element listy nazywamy *końcami listy*. Szczególnym przypadkiem listy może być **stos** (gdy pobrać, odczytać i wstawić element można tylko na końcu listy) lub **kolejka** (pobrać i odczytać element można tylko na początku listy, a dodać na końcu). W przeciwieństwie do stosu i kolejki listy mogą zawierać dwa wskaźniki. Takie listy nazywamy *dwukierunkowymi*, poruszać się możemy w niej w obydwu kierunkach, co przyspiesza wszystkie operacje.

Zalety : Małe zużycie pamięci, elastyczność

Wady : Niełatwe sortowanie, nienaturalny dostęp do elementów

Operacje na danych:

Insert(x,p) – wstaw x na pozycję p .

Locate(x) zwraca pozycję pierwszego wystąpienia elementu x .

Retrieve(p) – zwraca element występujący na pozycji p.

Delete(p) – usuwa element na pozycji p z listy.

Next(p) – zwraca pozycję następną w stosunku do p .

Previous(p)- zwraca pozycję następną w stosunku do p.

~Lista().Czyni listę pustą i zwraca pozycję END(L).

First() – zwraca pozycję pierwszego elementu.

printList() – wypisuje elementy w kolejności występowania.

Testy:

Przykładowy test znajduje się w kodzie(ewentualnie można odkomentować sekcję na dole pliku źródłowego, w celu testowania za pomocą pliku inputowego, jednak nie każdą metodę jest sens testować w ten sposób(np. zwracanie pozycji poprzednika - użytkownik nie ma z tej funkcji żadnego pożytku)

Wzorcowy output z testu w kodzie:

lista:

1

2

3

7

4

3

zlokalizowano: 2

adres next: zależnie od komputera

adres prev: zależnie od komputera

usunieto element

lista:

1

2

7

4

3

pozyskany numer: 7

poprzednik: 1