CANTHO UNIVERSITY

Chapter 2: Common abstract data types

Lâm Hoài Bảo - FSE – CICT
Trương Minh Thái – FSE - CICT

1

CANTHO UNIVERSITY

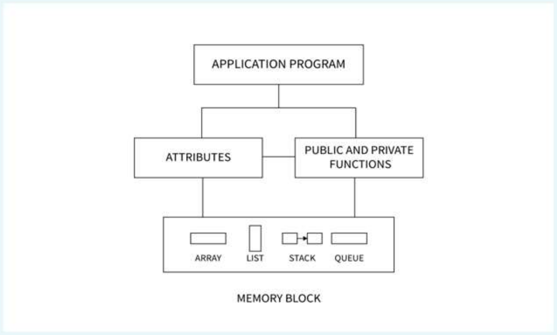
Objectives

- Understand abstract data types such as lists, stacks, and queues.
- Implement data types in C programming language.
- Apply abstract data types to real problems.

2

Content

- List abstract data type (LIST)
- Stack abstract data type (STACK)
- Queue abstract data type (QUEUE)



LIST

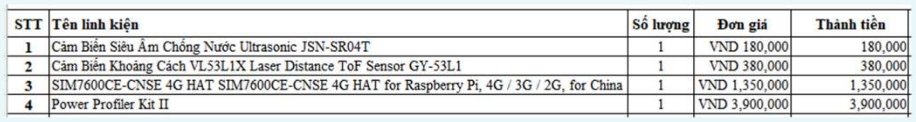
- List concept
- List operations
- List settings
  - Array-based list (ArrayList)
  - Using the cursor (Linked List)

List concept

- List of prime numbers<20

2	3	5	7	11	13	17	19

- List of equipments



STT	Tên linh kiện	Số lượng	Đơn giá	Thành tiền
1	Cảm Biến Siêu Âm Chống Nước Ultrasonic JSN-SR04T	1	VND 180,000	180,000
2	Cảm Biến Khoảng Cách VL53L1X Laser Distance ToF Sensor GY-53L1	1	VND 380,000	380,000
3	SIM7600CE-CNSE 4G HAT SIM7600CE-CNSE 4G HAT for Raspberry Pi, 4G / 3G / 2G, for China	1	VND 1,350,000	1,350,000
4	Power Profiler Kit II	1	VND 3,900,000	3,900,000

- A list is a finite set of elements of the same type
- The data type of elements in a list is called the element type.

www.ctu.edu.vn 5

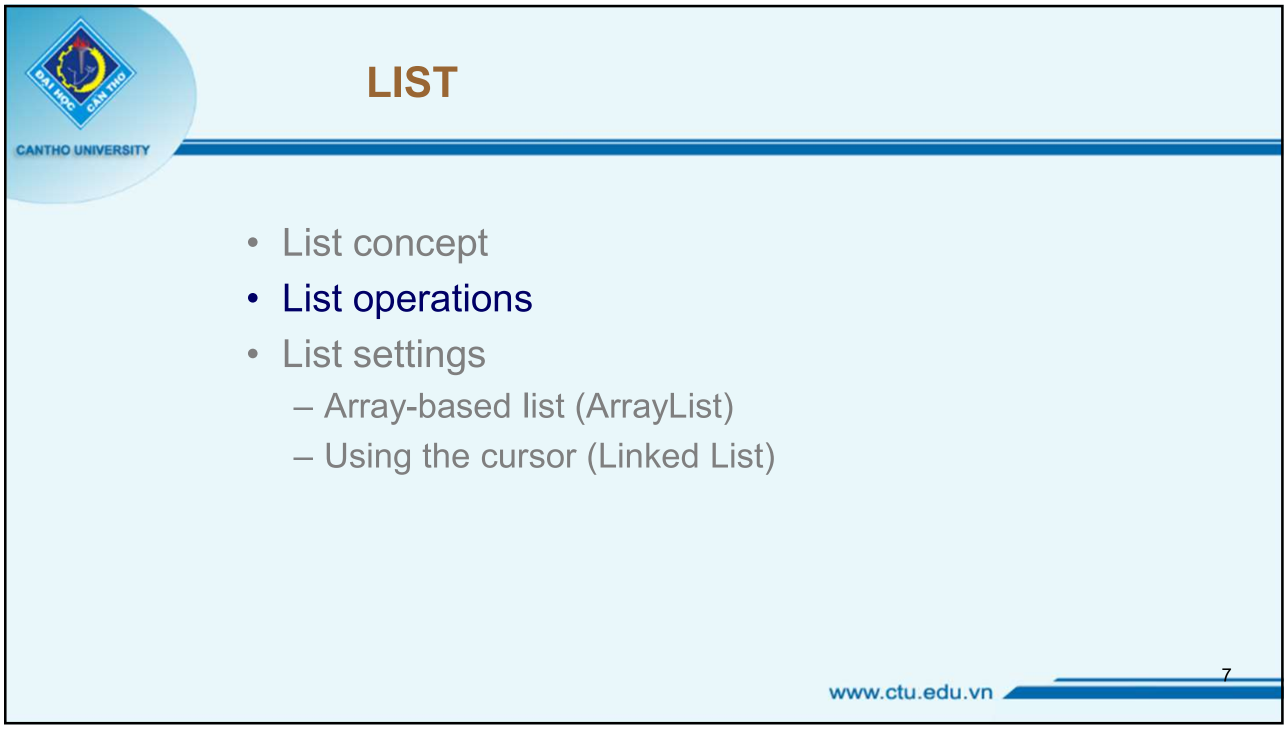
5

List concept

- A list is a finite set of elements of the same type
- The data type of an element in a list is called the element type.
- Length of the list: the number of elements of the list
- The elements in the list are in a linear order according to their position of occurrence, e.g. a_i before a_{i+1} (i=1..n-1)
- If
  - n=0: empty list
  - n>0: first element is a_1, last element is a_n

www.ctu.edu.vn 6

6

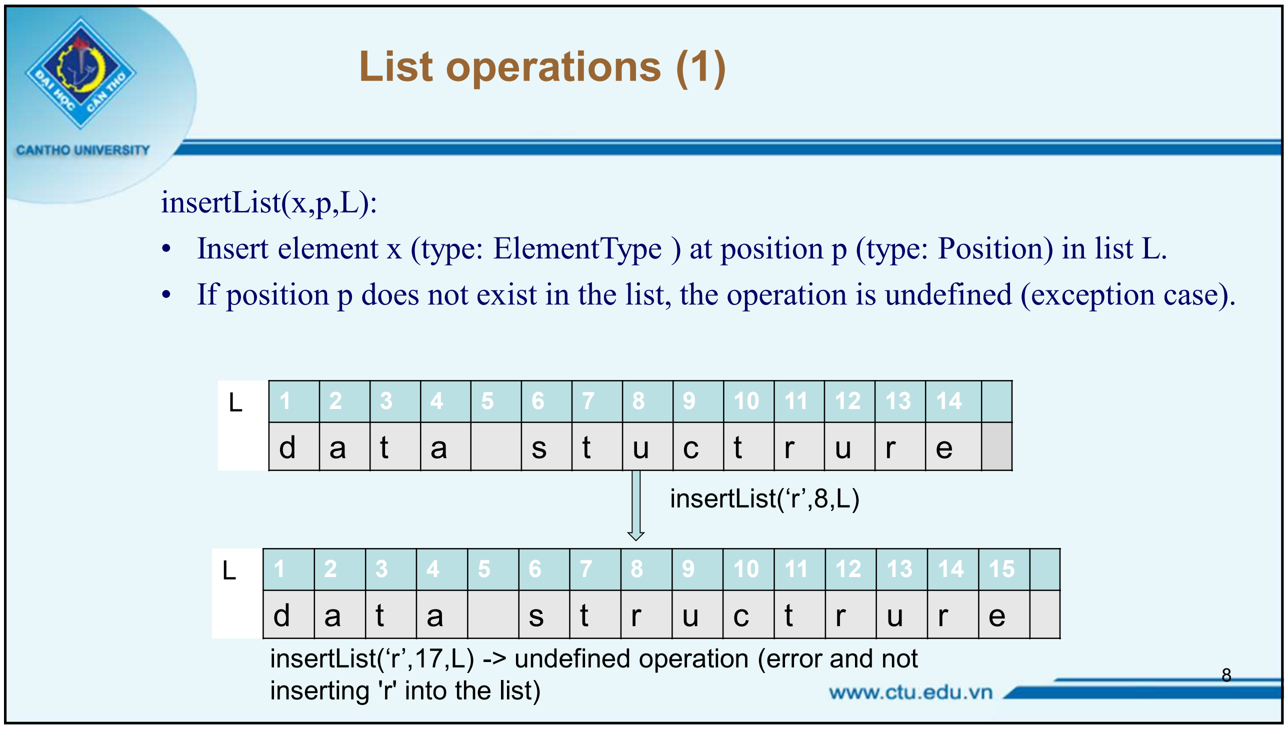


LIST

- List concept
- List operations
- List settings
  - Array-based list (ArrayList)
  - Using the cursor (Linked List)

www.ctu.edu.vn 7

7



List operations (1)

insertList(x,p,L):

- Insert element x (type: ElementType) at position p (type: Position) in list L.
- If position p does not exist in the list, the operation is undefined (exception case).

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	d	a	t	a		s	t	u	c	t	r	u	r	e	

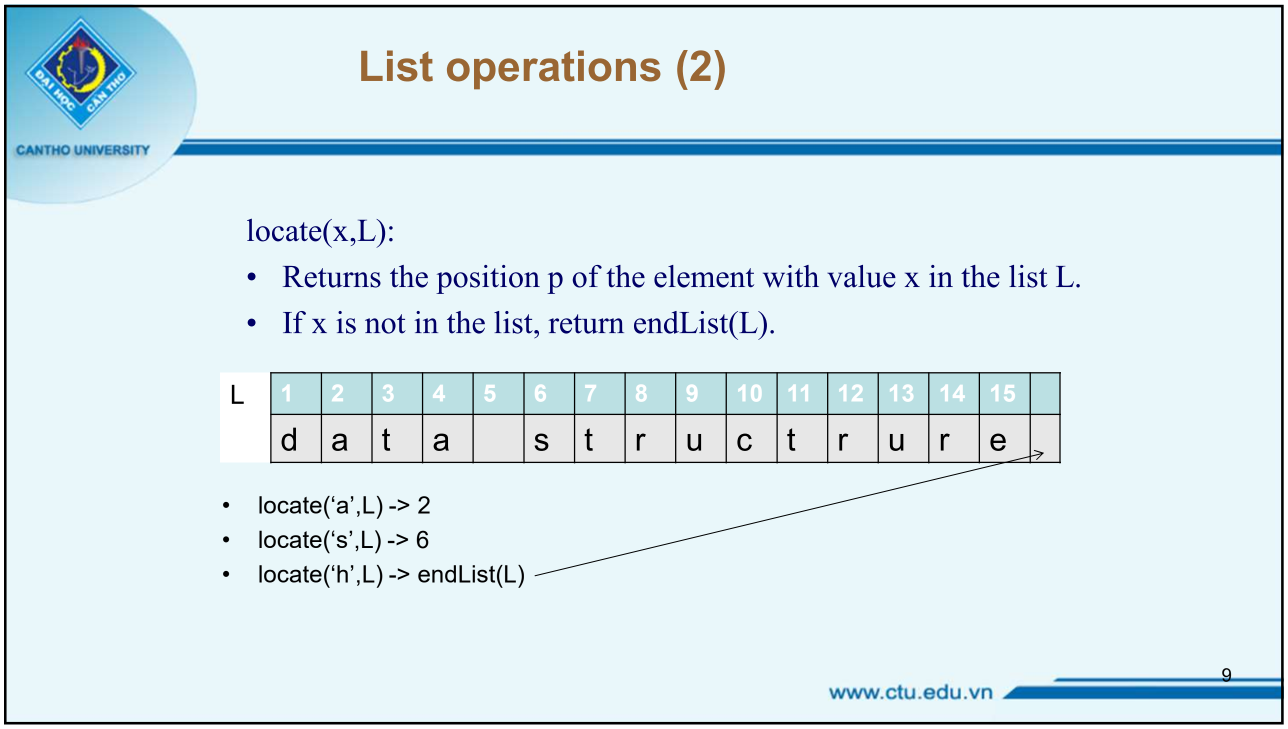
insertList('r',8,L)

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	d	a	t	a		s	t	r	u	c	t	r	u	r	e	

insertList('r',17,L) -> undefined operation (error and not inserting 'r' into the list)

www.ctu.edu.vn 8

8



List operations (2)

locate(x,L):

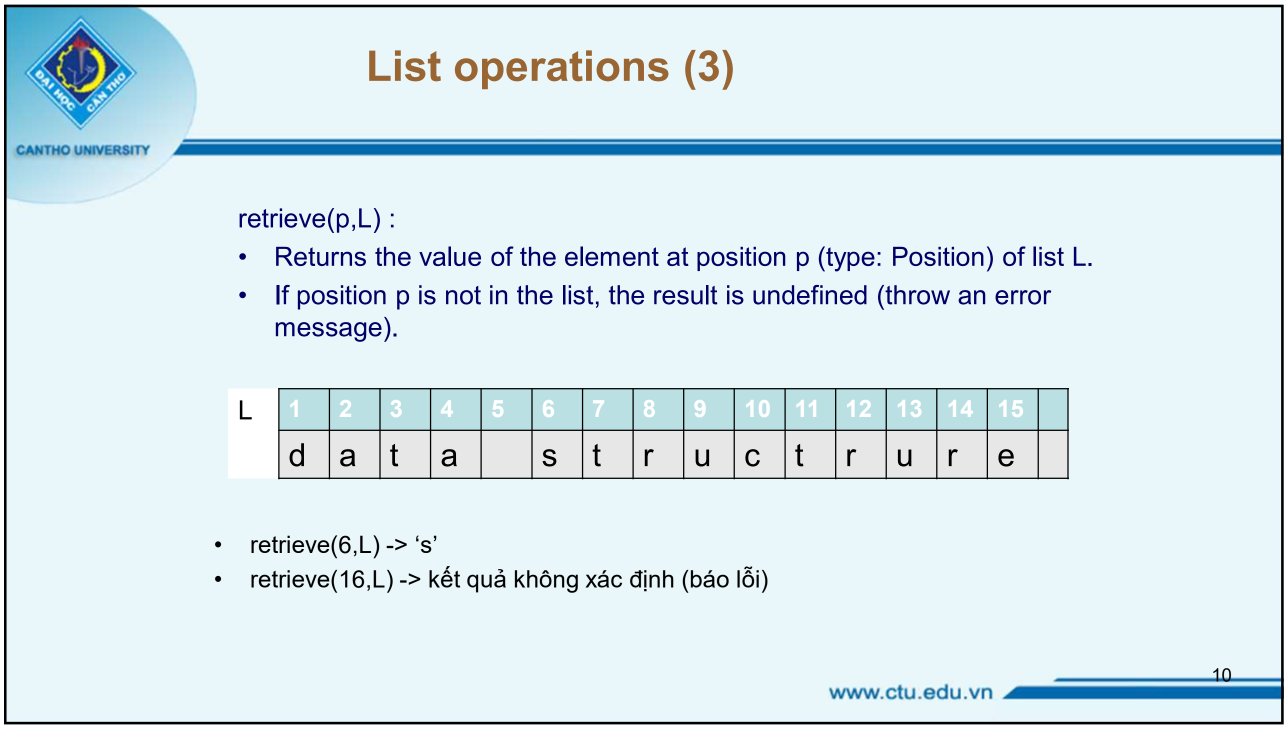
- Returns the position p of the element with value x in the list L.
- If x is not in the list, return endList(L).

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	d	a	t	a		s	t	r	u	c	t	r	u	r	e	

- locate(‘a’,L) -> 2
- locate(‘s’,L) -> 6
- locate(‘h’,L) -> endList(L)

www.ctu.edu.vn 9

9



List operations (3)

retrieve(p,L) :

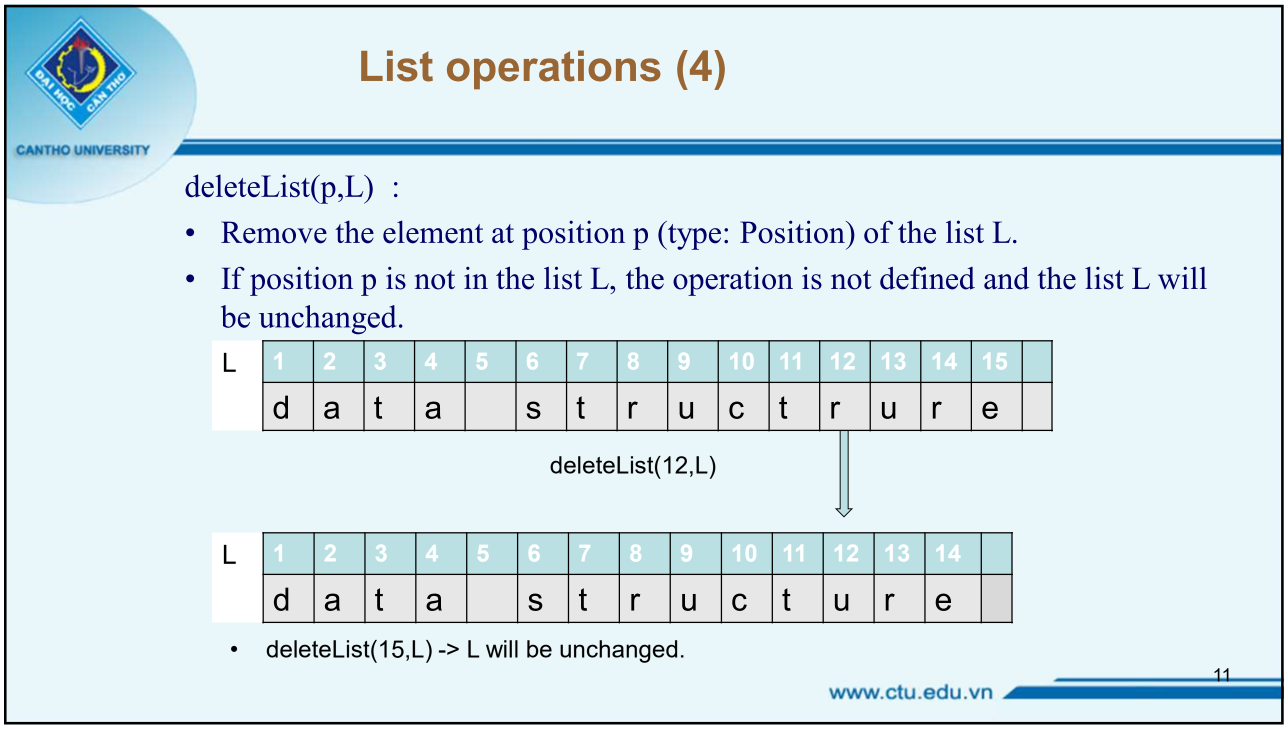
- Returns the value of the element at position p (type: Position) of list L.
- If position p is not in the list, the result is undefined (throw an error message).

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	d	a	t	a		s	t	r	u	c	t	r	u	r	e	

- retrieve(6,L) -> ‘s’
- retrieve(16,L) -> kết quả không xác định (báo lỗi)

www.ctu.edu.vn 10

10



List operations (4)

deleteList(p,L) :

- Remove the element at position p (type: Position) of the list L.
- If position p is not in the list L, the operation is not defined and the list L will be unchanged.

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	d	a	t	a		s	t	r	u	c	t	r	u	r	e	

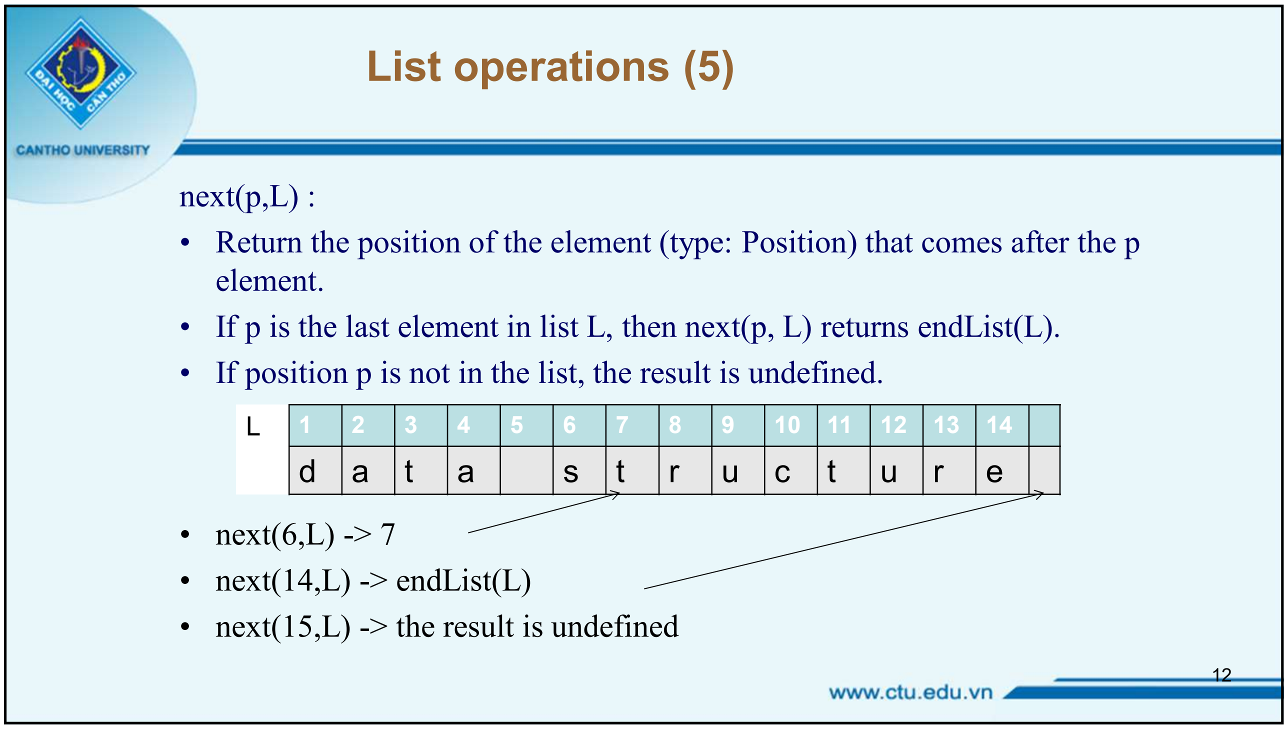
deleteList(12,L)

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	d	a	t	a		s	t	r	u	c	t	u	r	e	

- deleteList(15,L) -> L will be unchanged.

www.ctu.edu.vn 11

11



List operations (5)

next(p,L) :

- Return the position of the element (type: Position) that comes after the p element.
- If p is the last element in list L, then next(p, L) returns endList(L).
- If position p is not in the list, the result is undefined.

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	d	a	t	a		s	t	r	u	c	t	u	r	e	

- next(6,L) -> 7
- next(14,L) -> endList(L)
- next(15,L) -> the result is undefined

www.ctu.edu.vn 12

12

CANTHO UNIVERSITY

List operations (6)

previous(p,L) :

- Returns the position of the element before the element p.
- If p is the first element in the list, then previous(p, L) is undefined.
- If position p is not in the list, the result is also undefined.

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	d	a	t	a		s	t	r	u	c	t	u	r	e	

- previous(6,L) -> 5
- previous(1,L) -> kết quả không xác định
- previous(17,L) -> kết quả không xác định

www.ctu.edu.vn 13

13

CANTHO UNIVERSITY

List operations (7)

first(L) :

- Returns the position of the first element in the list L.
- If the list L is empty, endList(L) is returned.

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	d	a	t	a		s	t	r	u	c	t	u	r	e	

first(L) -> 1

L	

first(L) -> endList(L)

www.ctu.edu.vn 14

14

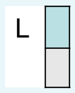
List operations (8)

emptyList (L) :

- Returns TRUE if the list L is empty. Otherwise, it returns FALSE.

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	d	a	t	a		s	t	r	u	c	t	u	r	e	

emptyList(L) -> FALSE

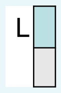


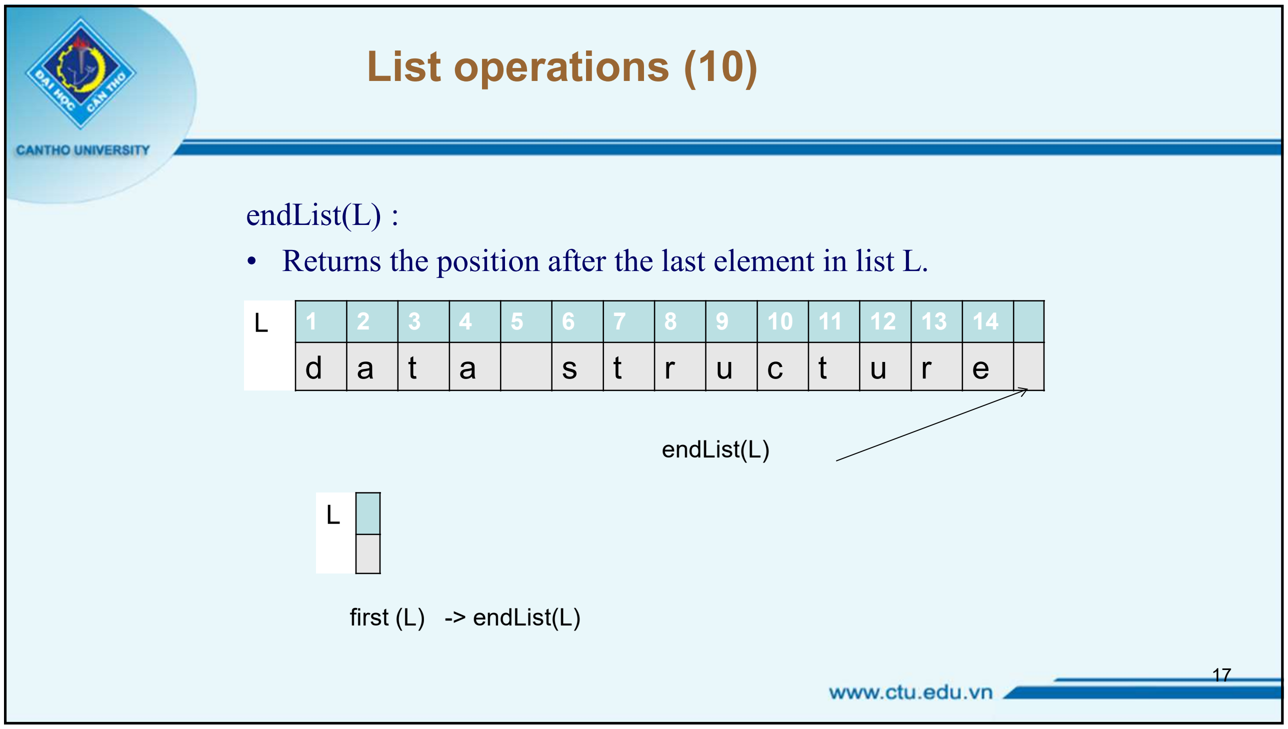
emptyList(L) -> TRUE

List operations (9)

makenullList(L) :

- Initialize an empty list L.





List operations (10)

endList(L) :

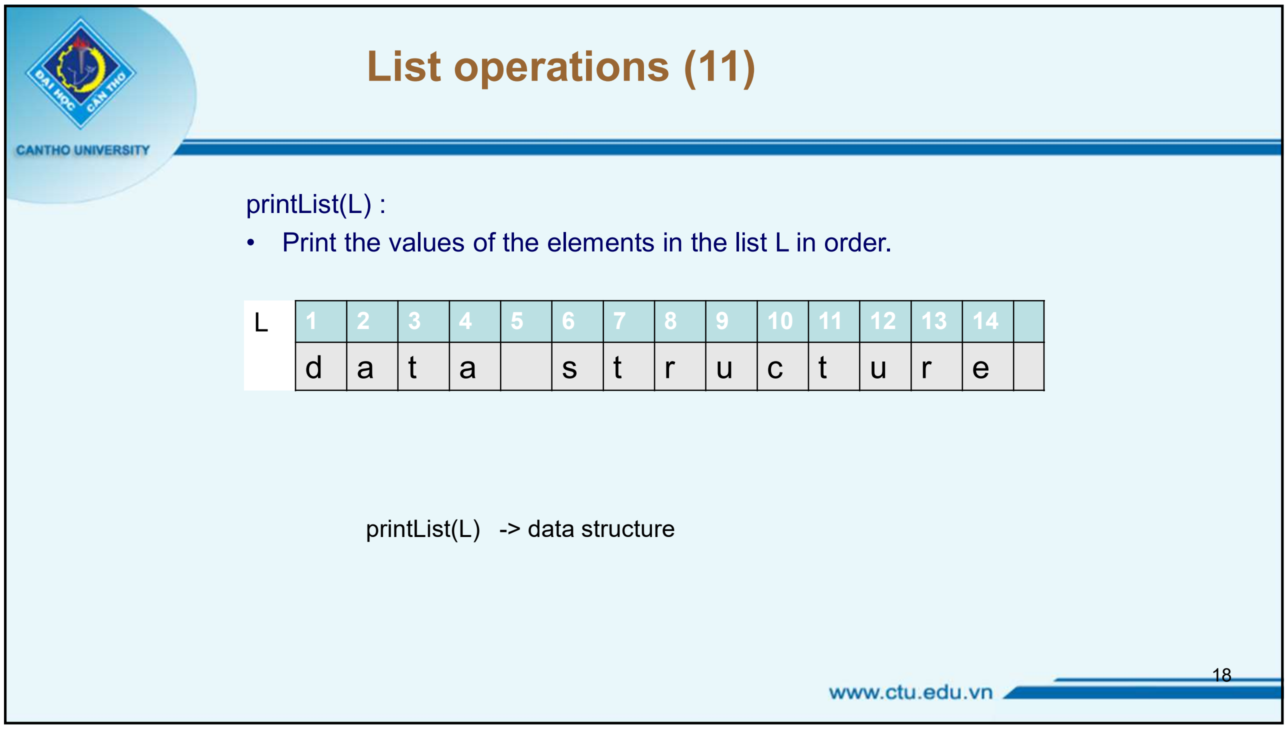
- Returns the position after the last element in list L.

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	d	a	t	a		s	t	r	u	c	t	u	r	e	

endList(L)

L	

first (L) -> endList(L)



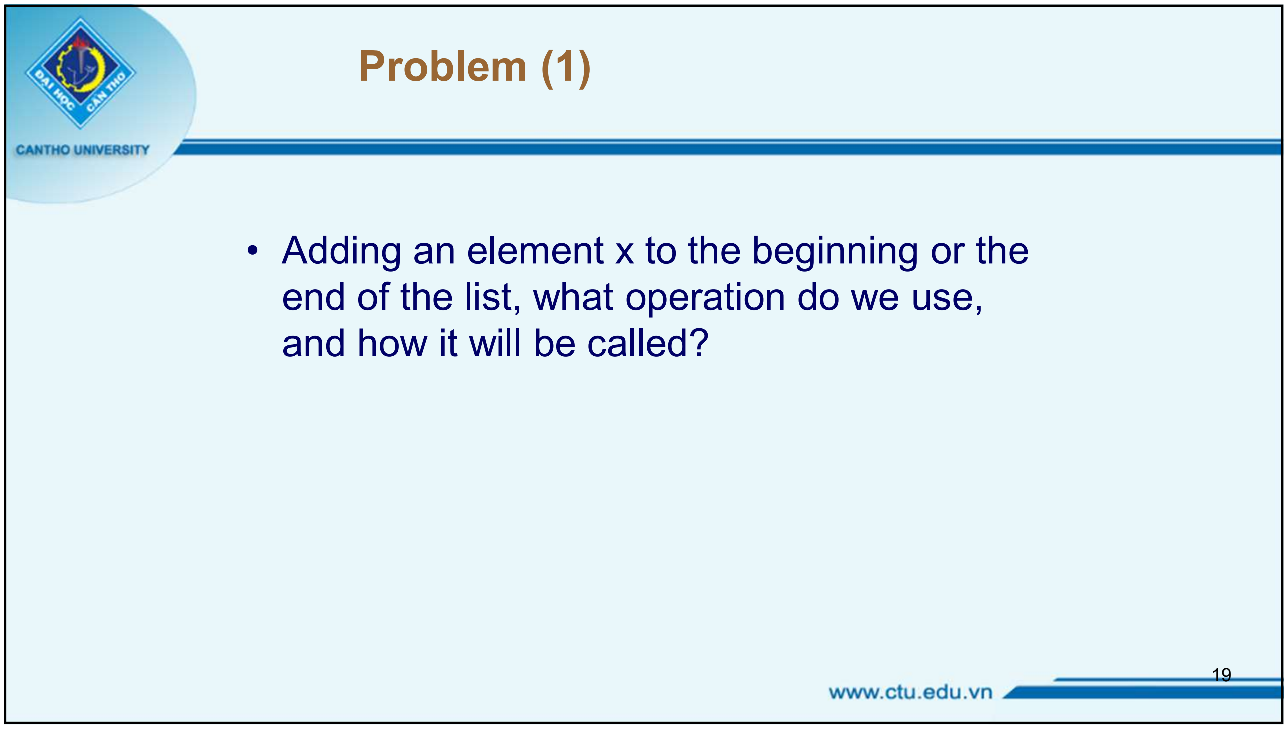
List operations (11)

printList(L) :

- Print the values of the elements in the list L in order.

L	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
	d	a	t	a		s	t	r	u	c	t	u	r	e	

printList(L) -> data structure



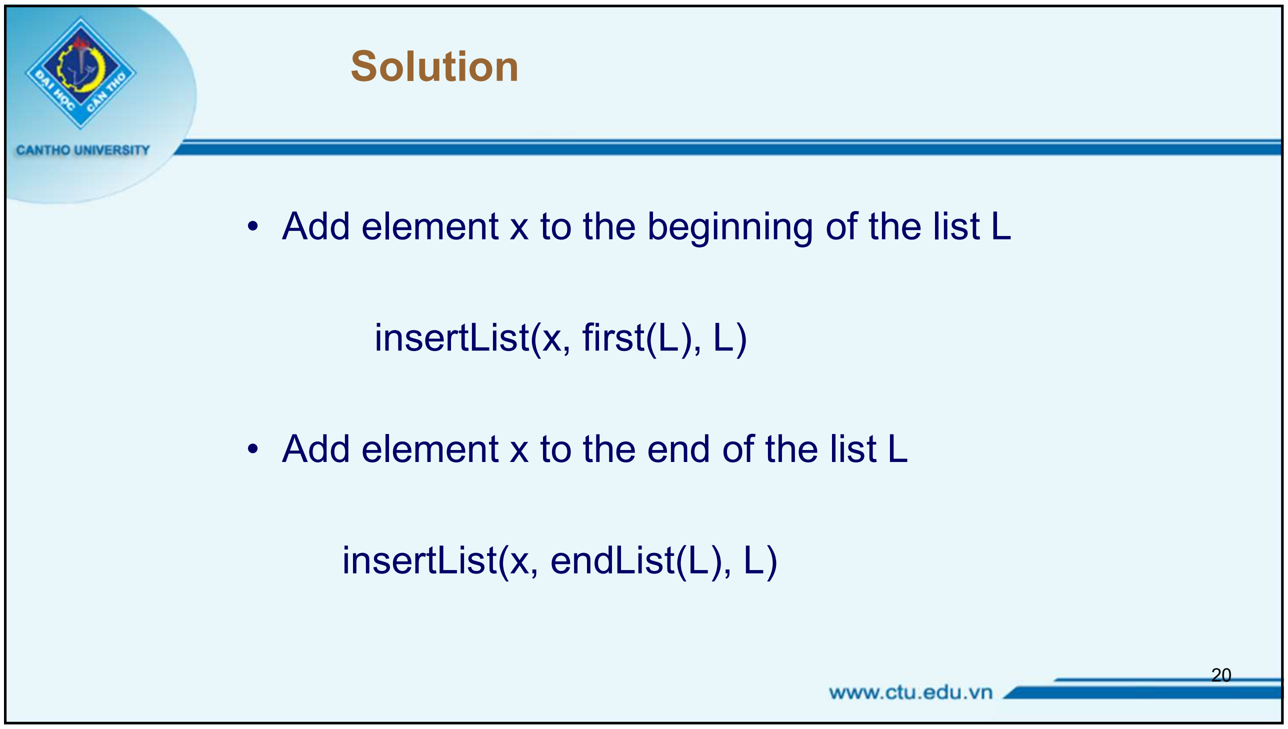
CANTHO UNIVERSITY

Problem (1)

- Adding an element x to the beginning or the end of the list, what operation do we use, and how it will be called?

www.ctu.edu.vn 19

19



CANTHO UNIVERSITY

Solution

- Add element x to the beginning of the list L

  insertList(x, first(L), L)

- Add element x to the end of the list L

  insertList(x, endList(L), L)

www.ctu.edu.vn 20

20

}

Problem (2)

CANTHO UNIVERSITY

Using abstract operations on lists, write a function that sorts the list in ascending order.

Solution

CANTHO UNIVERSITY

Using abstract operations on lists, write a function that sorts the list in ascending order.

```
void sort(List L){
        Position p,q;       //kiểu vị trí của các phần tử trong danh sách
        p= first(L);   //vị trí phần tử đầu tiên trong danh sách
        while (p!=endList(L)){
                q=next(p,L);//vị trí phần tử đứng ngay sau phần tử p
                while (q!=endList(L)){
                    if (retrieve(p,L) > retrieve(q,L))
                        swap(p,q);  // hoán đổi nội dung 2 phần tử
                   q=next(q,L);
                }
                p=next(p,L);
        }
}
```

Problem

- Using abstract operations on lists, write a delete_duplicate(LIST L) function that removes duplicate values in a list.
  - VD: L: data structure

$\xrightarrow{\text{delete\_duplicate(L)}}$ L:dat sruce

Solution

```
void delete_duplicate(List L)
{           Position p,q;        //kiểu vị trí của các phần tử trong danh sách
            p=first(L);  //vị trí phần tử đầu tiên trong danh sách
            while (p!=endList(L))
            {           q=next(p,L);       //vị trí phần tử đứng ngay sau phần tử p
                        while (q!=endList(L))
                        {
                          if (retrieve(p,L) == retrieve(q,L))
                              deleteList(q,L);  // xoa phần tử
                          else
                            q=next(q,L);
                      }
                      p=next(p,L);
            }
}
```

CANTHO UNIVERSITY

Array-based list

- Array implementation of List
- Operators
- Summary

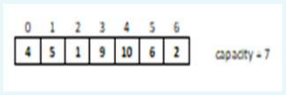
CANTHO UNIVERSITY

Array

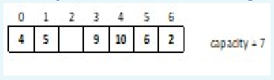
- Operate on a collection of elements
- In C:

```
int ids[7];  // array of 7 integers
float marks[40];
```

- Some issues
  - Fixed size



  - Modifications may cause discontiguous elements



ADT List

- A collection of elements of a given type (ElementType)
  - [A_1, A_2, …, A_n]
  - A_1 is at position 1, A_2 is at position 2, …
- Operators of a sequence DS
  - Add a new element
  - Remove an element
  - Access an element
  - …

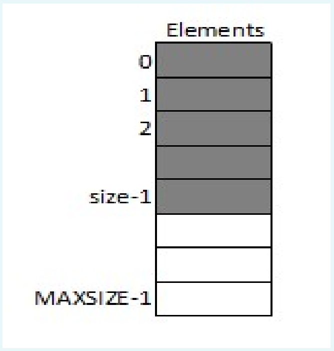
www.ctu.edu.vn 27

27

Array implementation

- An array to store elements
  - `elements`
- Estimate the maximum number of elements
  - `MAXSIZE`
- A variable to keep the current number of elements
  - `size`
- Position of each element is the index of that element:

  `[0.. size-1]`



www.ctu.edu.vn 28

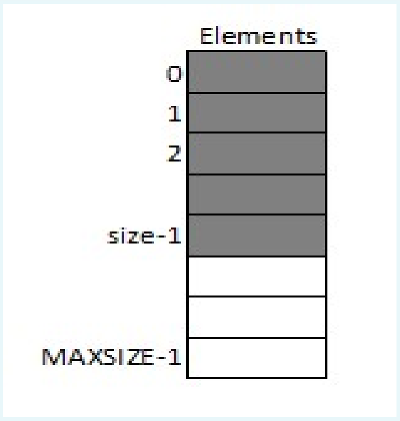
28

Declaration

```
#define MAXSIZE <number>
typedef <data type> ElementType;
typedef int Position;
typedef struct{
    ElementType elements[MAXSIZE];
    Position size;
}List;

List L;
```

- MAXSIZE: Maximum size of list
- elements: Array of elements
- size: current number of elements



Example [1]

- List of maximum of 10000 integers

```
#define MAXSIZE 10000
typedef int ElementType;
typedef struct{
    ElementType elements[MAXSIZE];
    int size;
}List;
```

Example [2]

- A polygon of maximum of 1000 vertices, each vertex is a pair of coordinates (x, y)

```
#define MAXSIZE 1000
typedef struct{
    int x, y;
}Point;
typedef Point ElementType;
typedef struct{
    ElementType elements[MAXSIZE];
    int size;
}List;
```

Example [3]

- A polygon of maximum of 1000 vertices, each vertex is a pair of coordinates (x, y)

```
#define MAXSIZE 1000
typedef struct{
    int x, y;
}Point;

typedef struct{
    Point vertices[MAXSIZE];
    int size;
}Polygon;
```

Array-based list

- Array implementation of List
- Operators
- Summary

www.ctu.edu.vn 33

33

List operators

Operator	Description
makeNull(&L)	Initialize an empty list
len(L)	Number of elements
empty(L)	Check whether the list is empty?
fullList(L)	Check whether the list is full?
print(L)	Traverse the list to print out all elements
getAt(p, L)	Return the element at position p
setAt(p, x, &L)	Update the element at position p by a new value x
insertAt(p, x, &L)	Insert x at position p
popAt(p, &L)	Remove and return the element at position p
insertFirst(x, &L)	Insert x to the first position
popFirst(&L)	Remove and return the first element
append(x, &L)	Append a new element to the list
popLast(&L)	Remove and return the last element
locate(x, L)	Return the position of the first appearance of x in the list

34

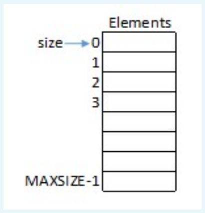
34

List construction

CANTHO UNIVERSITY

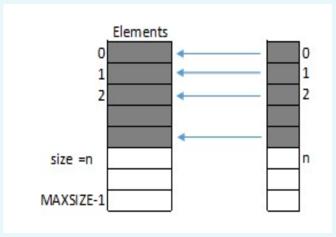
- Initialize an empty list
  - Set the current number of elements of the pointer of list is 0

```
ALGORITHM makeNull(*pL):
    pL->size ← 0
```



- Make a list from an array of elements
  - Copy each element of input array to the array element; then increase current number of elements of the list

```
ALGORITHM build(A, n, *pL):
    for i=0 to n-1:
        pL->Elements[i] <-- A[i]
    pL->size <-- n
```



www.ctu.edu.vn 35

Length of list
Check whether the list is empty?

CANTHO UNIVERSITY

- Length of list

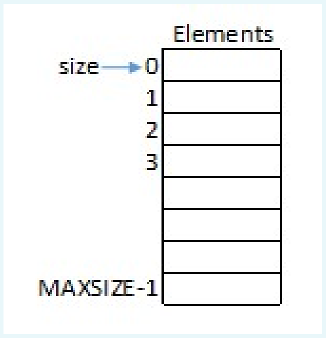
```
ALGORITHM len(L):
    return L.size
```

- Check list is empty?

```
ALGORITHM empty(L):
    return (L.size==0)
```

- Check list is full?

```
ALGORITHM full(L):
    return (L.size==MAXSIZE)
```



www.ctu.edu.vn 36

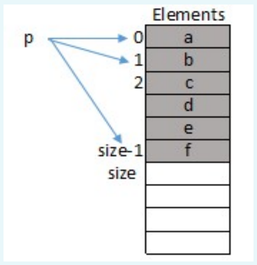
List traversal

- Visits each element of the list
- Algorithm

```
ALGORITHM traverse(L):
    for p=0 to len(L)-1:
        Process element at p
```

- Print the list to stdout

T(n) = O(n)



```
ALGORITHM print(L):
    for p=0 to len(L)-1:
        printf(get_at(p))
```

Get/Set

- Get the element at position p

```
ElementType getAt(int p, List L){
    return L.elements[p];
}
```

- Update element at position p

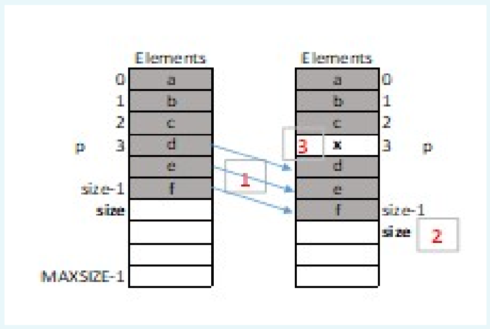
```
void setAt(ElementType x, int p, List *pL){
    pL->elements[p] = x;
}
```

- T(n) = O(1)

CANTHO UNIVERSITY

Insert element to position p

- Insert a new element at position p in the list pointed by the pointer L
  - `insertAt(x='x', p=3, &L)`
- Algorithm
  - If p is valid:
    - Right shift 1 position for each elements from size-1 - p
    - Increase the current size
    - Put x at position p
- Valid position: 0 … len()
  - 0: insert to the first of list
  - len(): append to the list



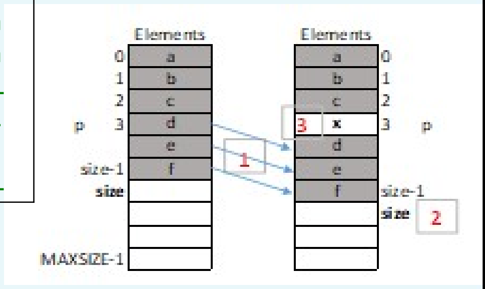
www.ctu.edu.vn 39

39

CANTHO UNIVERSITY

Insert new element at position p

```
ALGORITHM insertAt(x, p, *pL):
    if (p is valid):                                #1
        for q←pL->size to p+1:                      #n
            pL->elements[q] ← pL->elements[q-1]     #n
        pL->size ++                                 #1
        pL->elements[p] ← x                         #1
    else:
        RaiseError  "invalid position"              #1
```



$T(n) = 2n + 4$

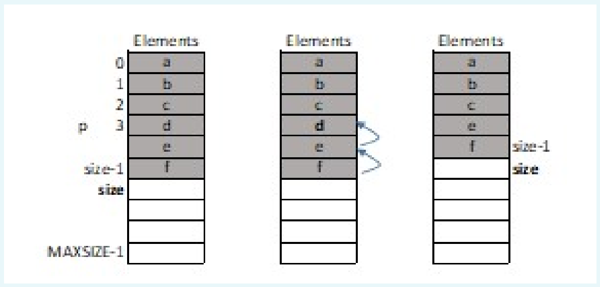
$\rightarrow T(n) = O(n)$

www.ctu.edu.vn 40

40

Remove and return the element at p

- Remove and return the element at position p pointed by the pointer L
  - popAt(p=3, &L) → d



www.ctu.edu.vn 41

41

Remove and return the element at p

- Algorithm
  - If p is valid
    - Shift left 1 position for each element from p+1 to size-1
    - Decrease the current size
    - Return the element at p
- Pseudo-code

```
ALGORITHM popAt(p, *pL):
    if p is valid:     #0 .. size-1
        x ← pL->Elements[p]
        for q ← p+1 to pL->size-1:
            pL->Elements[q-1] ← pL->Elements[q]
        pL->size--
        return x
    else:
        RaiseError “Invalid position”
        return ERROR
```

T(n) = O(n)

www.ctu.edu.vn 42

42

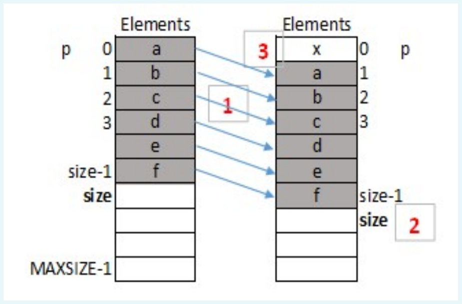
Insert to the first of list

- Insert a new element to the first of the list pointed by the pointer L
  - `insertFirst(x='x', &L)`

?

- `insertFirst()` is the worst case of `insertAt()`

  T(n) = O(n)

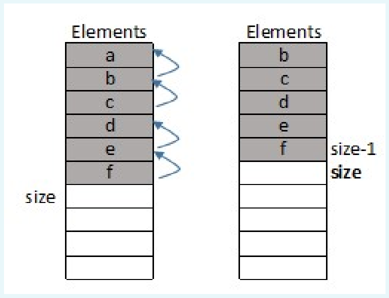


Remove and return the first element

- Remove and return the first element of the list pointed by the pointer L
  - `pop_first(&L)` → 'a'

?

- `popFirst()` is the worst case of `popAt()`:
  - T(n) = O(n)



'a'

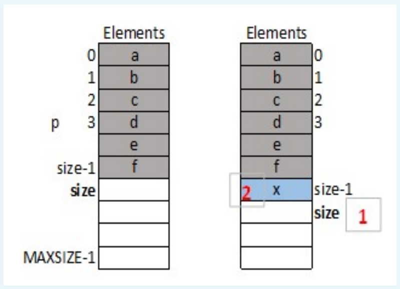
T(n) = O(n)

Append to the list

- Append a new element to the list pointed by the pointer L
  - `append(x='x', &L)`
- Pseudo-code

```
ALGORITHM append(x, *pL):
    pL->size++
    pL->elements[pL->size-1] ← x
```

- The best case of `insert_at()`
  - T(n) = O(1)



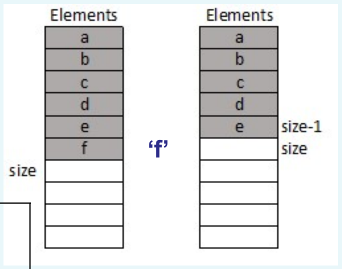
www.ctu.edu.vn

Remove and return the last element

- Remove and return the last element of the list pointed by the pointer L
  - `popLast(pL) → 'f'`
- Pseudo-code

```
ALGORITHM popLast(x, *pL):
    pL->size--
    return pL->elements[pL->size]
```

- The best case of `pop_at()`
  - T(n) = O(1)



www.ctu.edu.vn

Agenda

- Array implementation of List
- Operators
- Summary

Summary

- List implementation using array
- Time complexity in the worst case

Data Structures	Construction	Static	Dynamic		

	build()	get_at()
		set_at()
			insert_at()
			pop_at()
				insert_first()
				pop_first()
					append()
					pop_last()

Array	O(n)	O(1)	O(n)	O(n)	O(1)

CANTHO UNIVERSITY

Q&A

www.ctu.edu.vn

49