A struct can contain a pointer

```
struct ListElement
{
  RECT aRect;              // RECT member of structure
  ListElement* pNext;      // Pointer to a list element
};
```

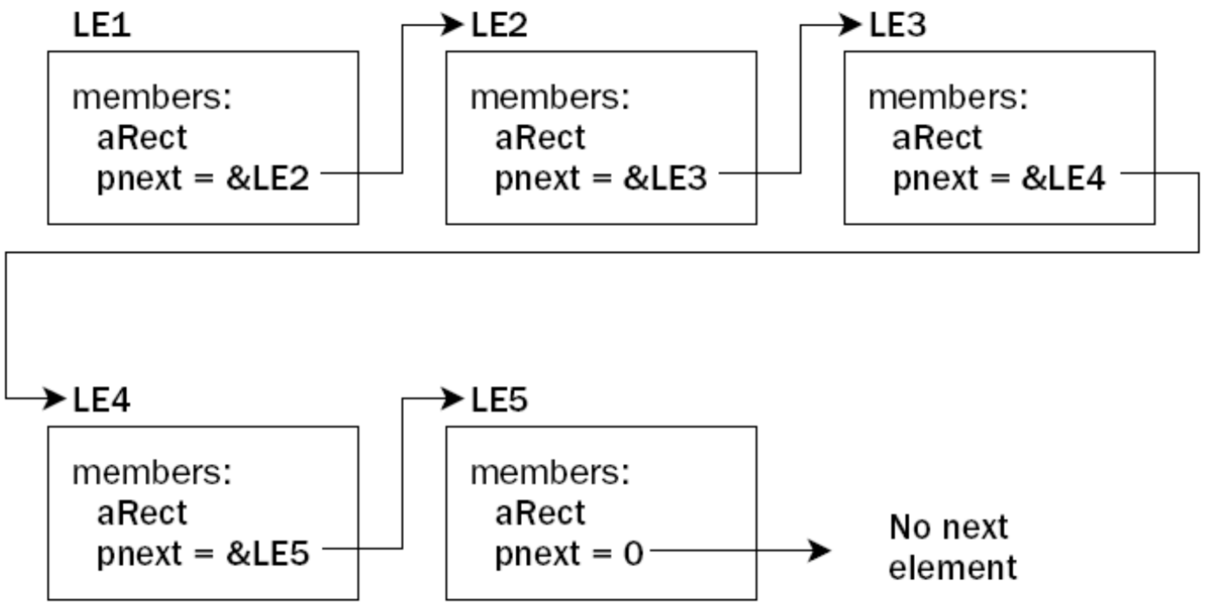
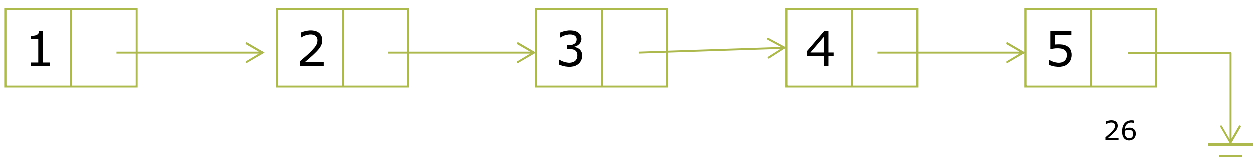


Figure 7-3 Linked List

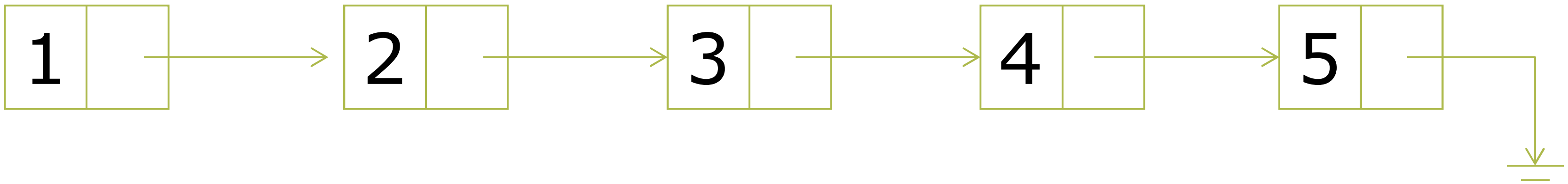
Create a Linked List

```
struct ListElement
{
  int value;                 // value of an element
  ListElement* pNext;        // Pointer to a list element
};
int main()
  {
      ListElement LE5 = { 5, NULL };
      ListElement LE4 = { 4, &LE5 };
      ListElement LE3 = { 3, &LE4 };
      ListElement LE2 = { 2, &LE3 };
      ListElement LE1 = { 1, &LE2 };
      PrintList(&LE1);
      return 0;
  }
```



Print a Linked List

```
void PrintList(ListElement* p)
{
    while (p != NULL)
    {
        std::cout << p->value;
        p = p->pNext;
    }
}
```



Dynamic Memory Allocation (P.201)

- Sometimes depending on the input data, you may allocate different amount of space for storing different types of variables at execution time

```
int n = 0;
cout << "Input the size of the vector - ";
cin >> n;
int vector[n];
```

error C2057: expected constant expression

Why Use Pointers? (P.183)

- Use pointer notation to operate on data stored in an array
- Enable access within a function to arrays, that are defined outside the function
- Allocate space for variables dynamically.

Free Store (Heap)

- To hold a string entered by the user, there is no way you can know in advance how large this string could be.

- Free Store - When your program is executed, there is unused memory in your computer.
- You can dynamically allocate space within the free store for a new variable.

The new Operator

- Request memory for a double variable, and return the address of the space
  - `double* pvalue = NULL;`
  - `pvalue = new double;`
- Initialize a variable created by new
  - `pvalue = new double(9999.0);`
- Use this pointer to reference the variable (indirection operator)
  - `*pvalue = 1234.0;`

The delete Operator

- When you no longer need the (dynamically allocated) variable, you can free up the memory space.
  - `delete pvalue;`
    - Release memory pointed to by pvalue
  - `pvalue = NULL;`
    - Reset the pointer to NULL

- After you release the space, the memory can be used to store a different variable later.

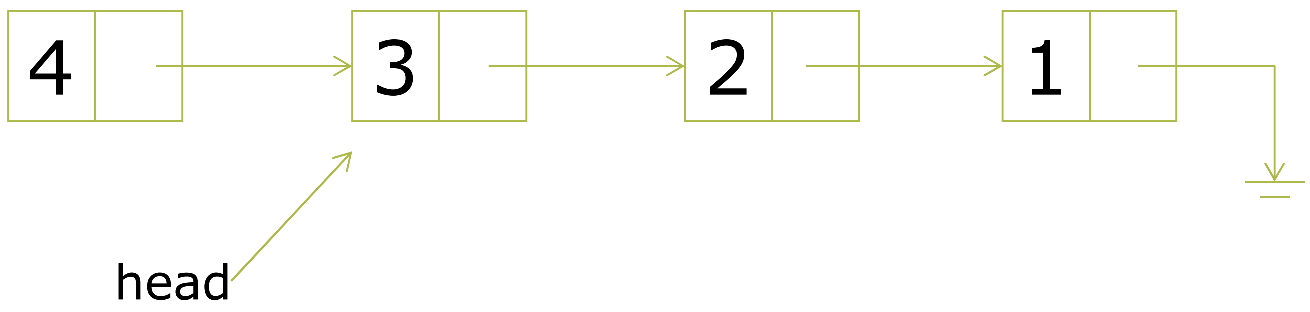
Allocating Memory Dynamically for Arrays

- Allocate a string of twenty characters
  - `char* pstr;`
  - `pstr = new char[20];`
  - `delete [] pstr;`
    - Note the use of square brackets to indicate that you are deleting an array.
  - `pstr = 0;`
    - Set pointer to null

Exercise to Upload

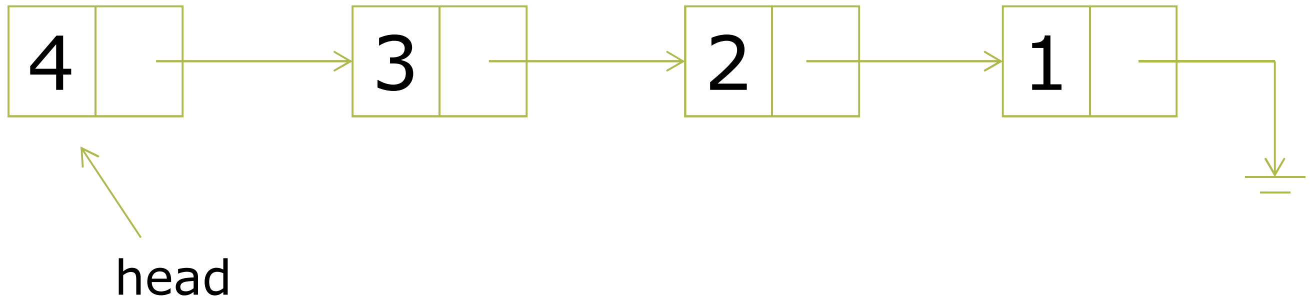
1. Write a program to read a series of positive integers from the user. The total number of input is unknown. Stop when the user supplies 0 or a negative number. Then output the series of numbers in reserve order.
   - For example, the input is 1 3 5 7 2 4 6 0,
     the output will be 6 4 2 7 5 3 1.
   - Hint: Store the input numbers in a linked list.

Adding a New Element



- Allocate a new element to store the input value.
- Update LE4.pnext to point to LE3.
- Update head pointing to LE4.

Adding a New Element



- Allocate a new element to store the input value.
- Update LE4.pnext to point to LE3.
- Update head pointing to LE4.