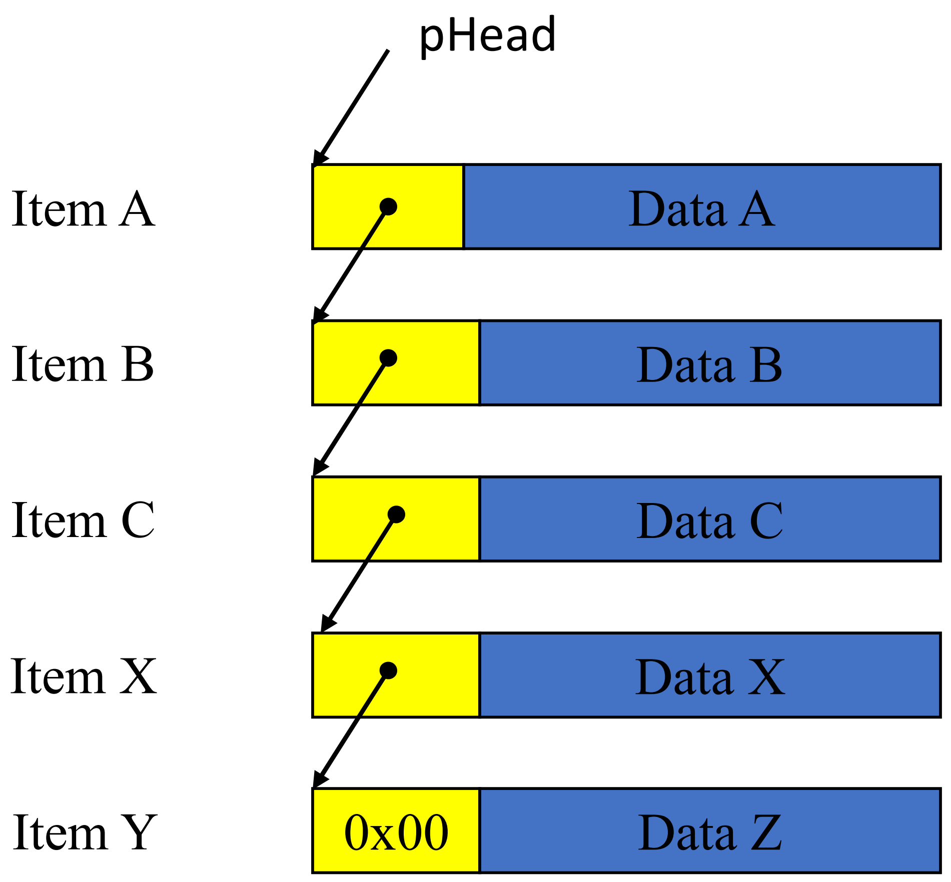
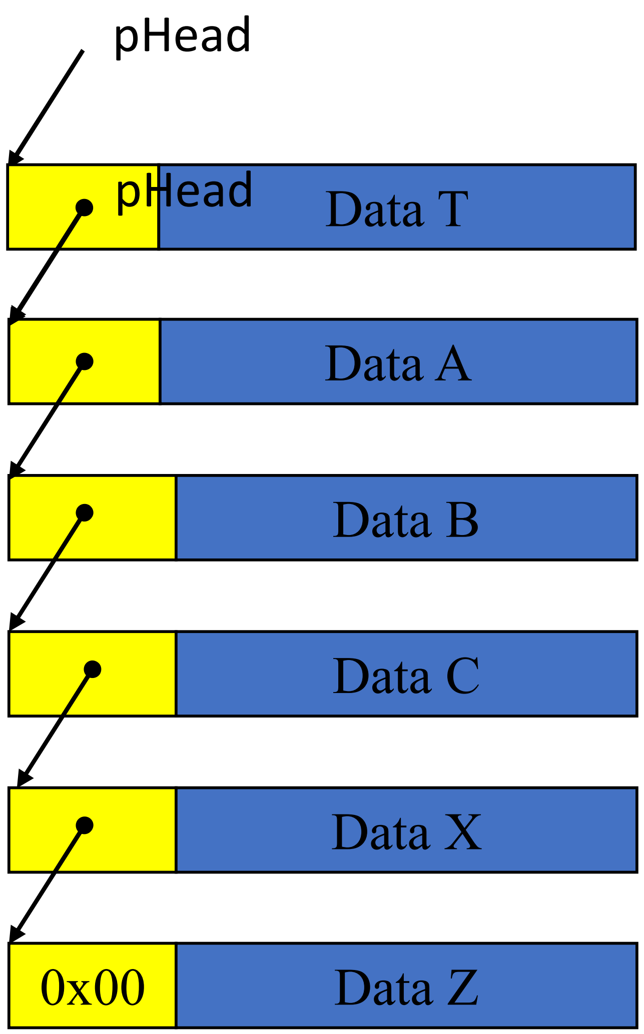
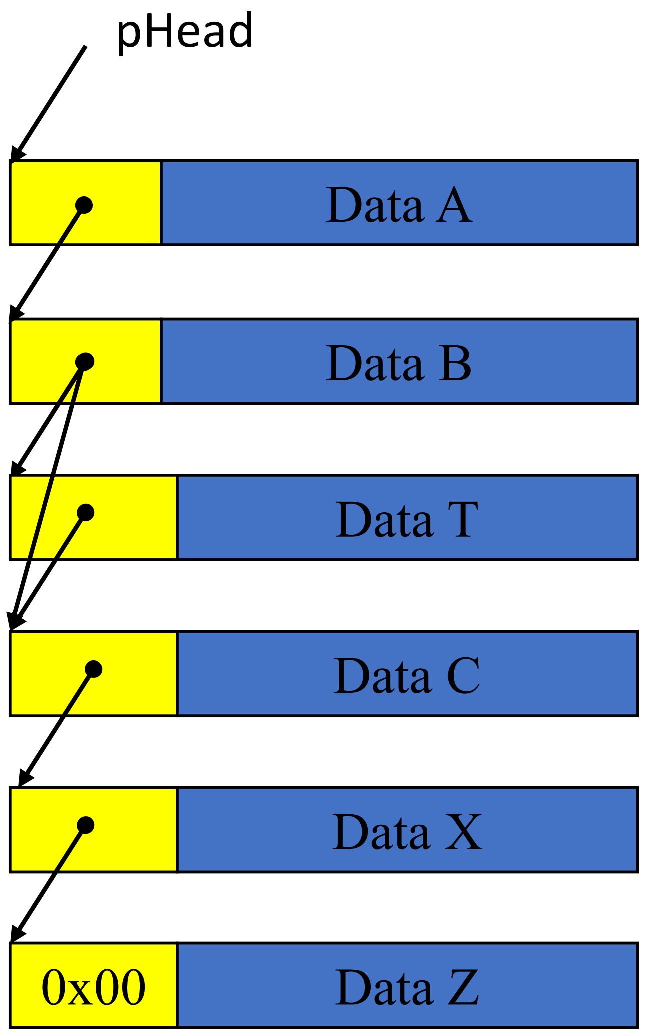
4.4.1 Linked list



Linked list: Insert data

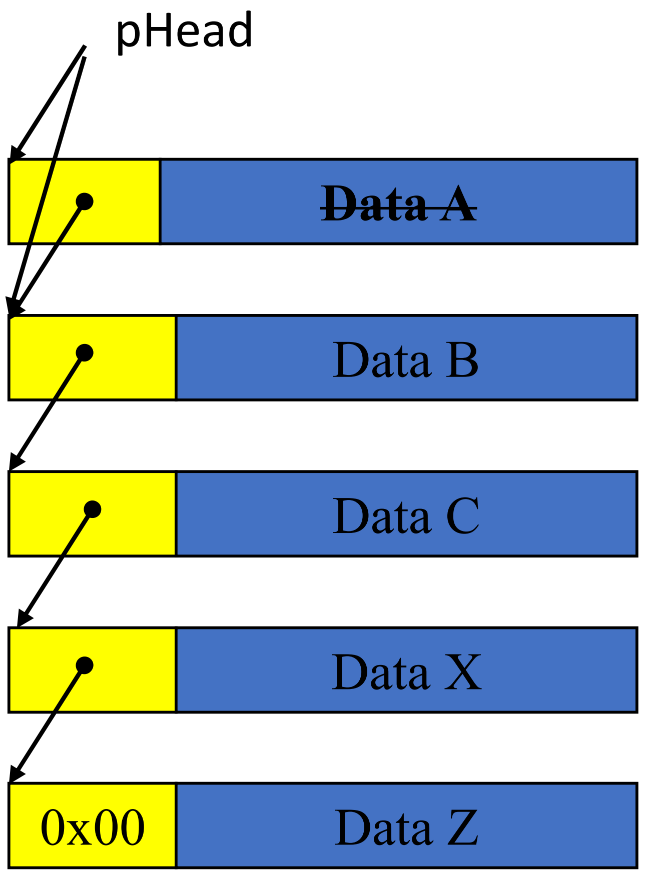


At the beginning of the list

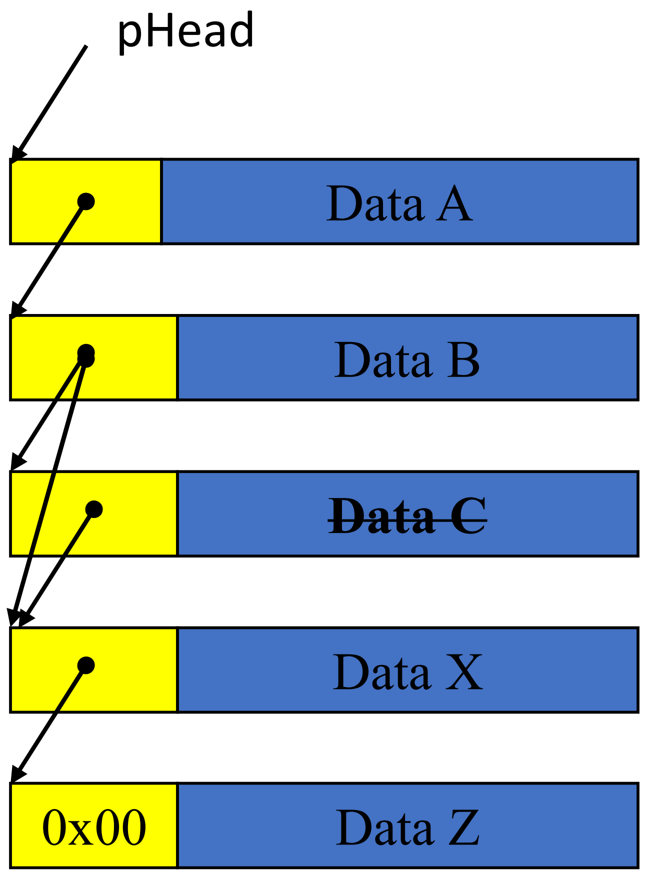


In the middle of the list

Linked list: Delete data



At the beginning of the list



In the middle of the list

Summary

- Advantages:
  - Flexible usage, allocating memory when needed and deallocating after using
  - Add/delete element via pointer; time taken to perform these task is constant, doesn't depend on data length or position
  - Access data in sequence
- Disadvantages:
  - Added element must be allocated dynamic memory
  - Deleting element requires respected memory space to be freed
  - If data type is not large, the overhead may be dominant
  - Searching data is based on linear methods which consume more time

Example: mail box

```
#include <string>
using namespace std;
struct MessageItem {
  string subject;
  string content;
  MessageItem* pNext;
};
struct MessageList {
 MessageItem* pHead;
};
void initMessageList(MessageList& l);
void addMessage(MessageList&, const string& sj,
                              const string& ct);
bool removeMessageBySubject(MessageList&l,
                              const string& sj);
void removeAllMessages(MessageList&);
```

Example: mail box (cont.)

```
#include "List.h"
void initMessageList(MessageList& l) {
   l.pHead = 0;
}
void addMessage(MessageList& l, const string& sj,
                                constsstring& ct) {
   MessageItem* pItem = new MessageItem;
   pItem->content = ct;
   pItem->subject = sj;
   pItem->pNext = l.pHead;
   l.pHead = pItem;
}
void removeAllMessages(MessageList& l) {
  MessageItem *pItem = l.pHead;
  while (pItem != 0) {
      MessageItem* pItemNext = pItem->pNext;
      delete pItem;
      pItem = pItemNext;
  }
  l.pHead = 0;
}
```

Example: mail box (cont.)

```
bool removeMessageBySubject(MessageList& l,
                              conststring& sj) {
  MessageItem* pItem = l.pHead;
  MessageItem* pItemBefore;
  while (pItem != 0 && pItem->subject != sj) {
      pItemBefore = pItem;
      pItem = pItem->pNext;
  }
  if (pItem != 0) {
  if (pItem == l.pHead)
     l.pHead = 0;
  else
     pItemBefore->pNext = pItem->pNext;
  delete pItem;
  }
  return pItem != 0;
}
```

Example: mail box usage (cont.)

```
#include <iostream>
#include "list.h"
using namespace std;
void main() {
  MessageList myMailBox;
  initMessageList(myMailBox);
  addMessage(myMailBox,"Hi","Welcome, my friend!");
  addMessage(myMailBox,"Test","Test my mailbox");
  addMessage(myMailBox,"Lecture Notes","Programming Techniques");
  removeMessageBySubject(myMailBox,"Test");
  MessageItem* pItem = myMailBox.pHead;
  while (pItem != 0) {
      cout << pItem->subject << ":" << pItem->content << '\n';
      pItem = pItem->pNext;
  }
  char c;
  cin >> c;
  removeAllMessages(myMailBox);
}
```

Homework

- Create a linked-list consisting of public holidays of a year and description of each day (as string), so that
  - A new public holiday can be added to the beginning of the list
  - Search for the description of the day (input argument is a date including day and month)
  - Delete a public holiday at the beginning of the list
  - Delete a public holiday in the middle of the list (input argument is a date including day and month)
  - Clear the whole list
- Write a program to demonstrate the usage of the above list