《面向对象程序设计》之——

链　表

连　盛

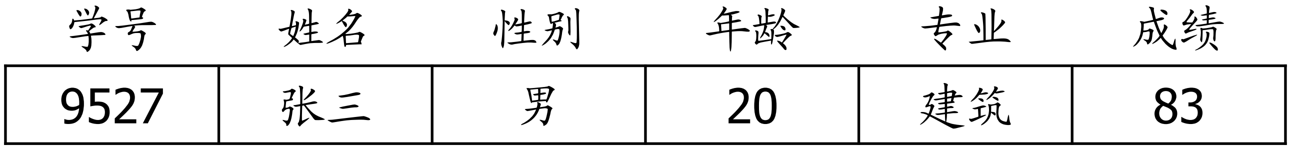
shenglian@fzu.edu.cn

计算机与大数据学院

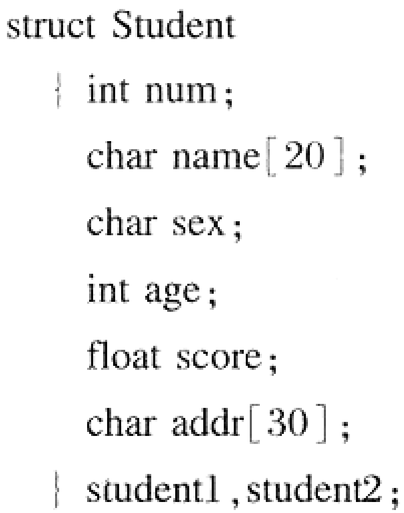
福州大学

链表：相关概念回顾

- 结构：一种用户自定义类型
- 对数组的扩展：
  - 数组中各元素是同一数据类型
  - 结构可以将不同类型的数据组合成有机整体。



学号	姓名	性别	年龄	专业	成绩
9527	张三	男	20	建筑	83



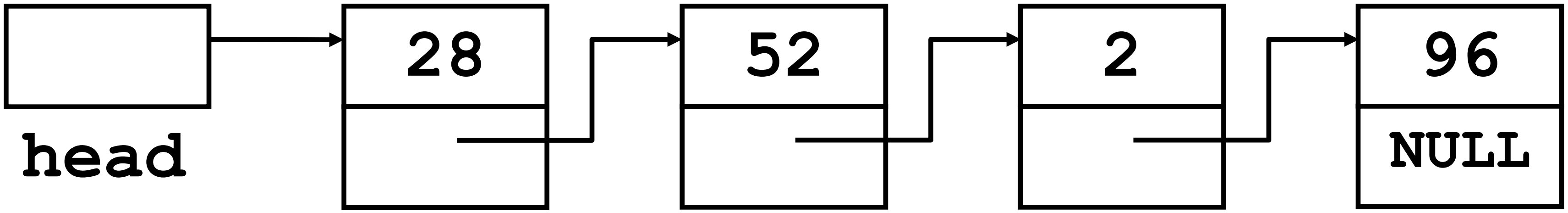
```
struct Student
  { int num；
    char name[20]；
    char sex；
    int age；
    float score；
    char addr[30]；
  } student1，student2；
```

链表的基本概念

- 结构数组--必须将数组的大小设定成足够大的值
  - 太浪费
  - 能否需要多少分配多少?
- 链表 = 动态内存分配 + 结构 + 指针
  - 若干同类型自引用结构（被称为结点）形成一条链
  - 可以在任何地方插入或删除结点

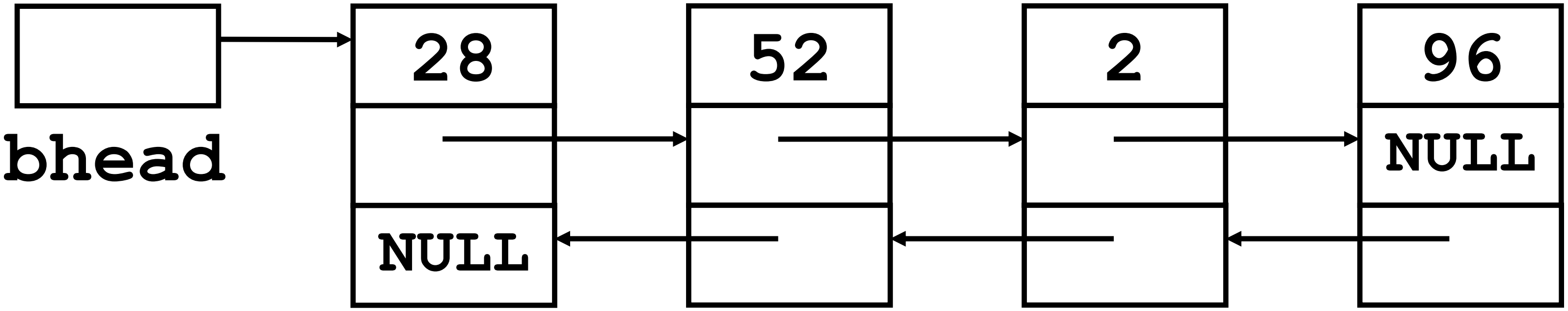
自引用结构及单向链表举例

```
struct node
 { int data;
   node * next;
 };
node *head;
```



自引用结构及双向链表举例

```
struct bnode
{ int data;
   bnode * next;   //指向后续结点
   bnode * pre;    //指向前面结点
};
bnode *bhead;
```



链表的建立

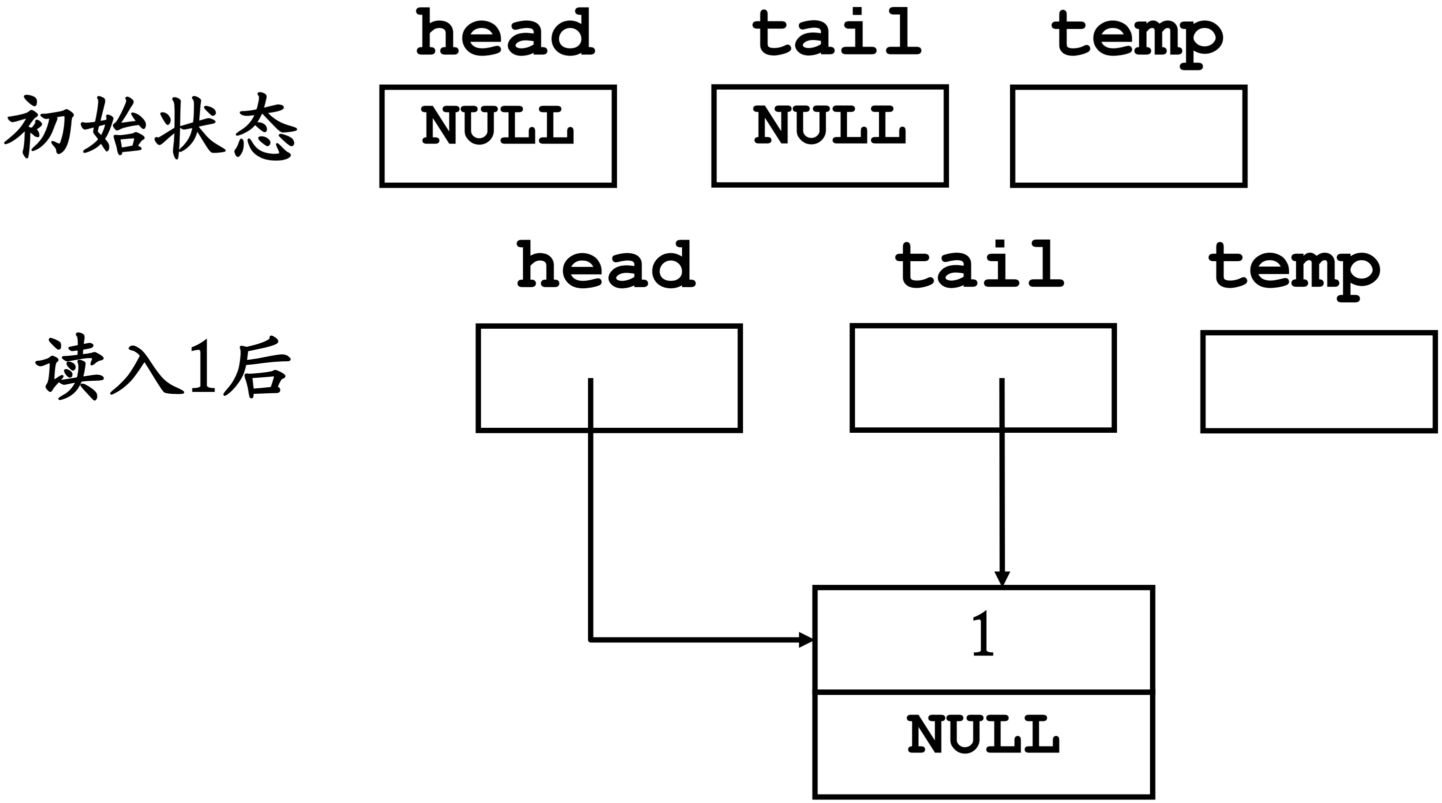
- 定义表示结点的结构类型
- 声明一个链首指针变量（如head），并赋初值NULL(包含0个结点的链表)
- 利用动态内存分配生成一个新结点，将该结点插入链尾、链头或链中
- 重复上一步

例子1：建立单向链表，读入n个整数，每个整数形成一个新结点插入到链尾

```
#include <iostream>
struct node
 { int data; node * next; };
node * createList ( int n );
main( )
{ int n;
  node * listHead = NULL;
  cout<<"Please enter the number of nodes:";
  cin >> n;
  if (n > 0)
    listHead = createList(n);
  return 0;
}
```

```
node *createList( int n )
{ node *temp,*tail = NULL,*head = NULL ;
  int num;
  cin >> num;
  head = new node ;  // 为新结点动态分配内存
  if (head == NULL)
   { cout << "No memory available!";
     return NULL;
   }
  else
   { head -> data = num;
     head -> next = NULL;
     tail = head;
   }
  for ( int i = 0; i < n - 1; i++ )
  { cin >> num;
    temp = new node ;// 为新结点动态分配内存
    if (temp == NULL)
    {cout << "No memory available!";
     return head;
    }
   else
    {temp->data = num;
     temp->next = NULL;
     tail->next = temp;
     tail = temp;
    }
  }
 return head ;
}
```

建立单向链表过程



head tail temp

初始状态

NULL NULL

head tail temp

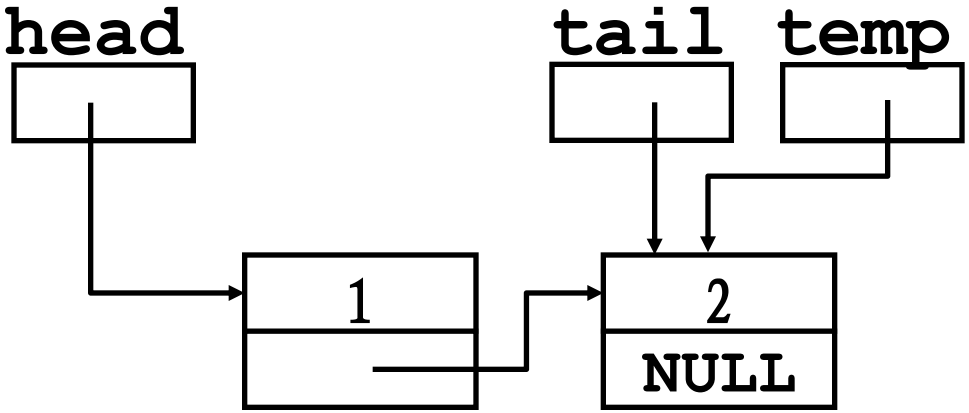
读入1后

1

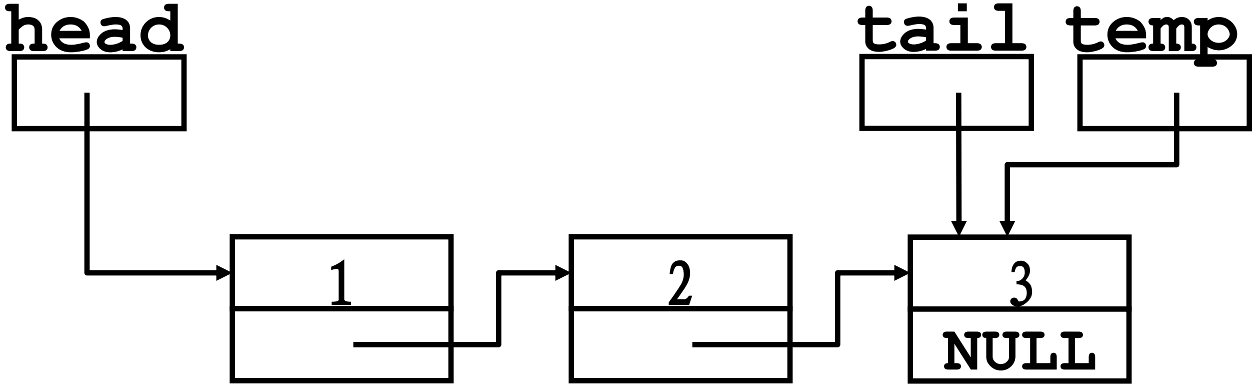
NULL

建立单向链表过程

读入2后



读入3后



链表的典型操作

- 遍历链表
  - 依次访问链表中的每个结点的信息
- 在链表中结点a之后插入结点c
- 从链表中删除一个结点c

  (1) 在链表中查找要删除的结点c;

  (2) 如果c有前驱结点(设为p)，则将p的后继指针指向c的后继节点：p->next=c->next

  (3) 释放c占用的空间

例子2：编写一个函数，输出例1链表中各结点的data成员的值

```
void outputList( node * head )
{ cout << "List: ";
  node *curNode = head;
  while ( curNode )
  { cout << curNode->data;
    if (curNode ->next)
        cout << " -> ";
    curNode = curNode ->next;
  }
  cout << endl;
  return;
}
```

例子3：编写一个函数，在例1的链表中查找包含指定整数的结点

```
node * findData(int n, node * head)
{node *curNode = head;
 while ( curNode )
 {if ( curNode->data == n)
  {cout<<"Find "<<n<<" in the list.\n";
   return curNode;
  }
  curNode = curNode->next;
 }
 cout<<"Can't find "<<n<<endl;
 return NULL;
}
```

例子5：编写一个函数，删除例1的链表中包含指定整数的结点

```
node *deleteData(int n,node * head)
{node *curNode = head;// 指向当前结点
 node *preNode = NULL;//指向当前结点的前驱结点
 while ( curNode )
  循环体
 cout<<"Can't find “
     <<n<<" in the list."<<endl;
 return head;
}
```

```
//  while ( curNode ) 的循环体：
  {if ( curNode->data == n)
   {if (preNode == NULL)
     head = head->next;
    else
     preNode->next = curNode->next;
    delete curNode;
    cout<<"Delete "<<n<<endl;
    return head; // 返回链首指针
   }
   preNode = curNode; // 当前结点变为前驱结点
   curNode = curNode->next;
  }
```

例子6：编写一个函数，按数据输入的顺序为n个整数建立双向链表

```
bnode *createBidirList (int n)
{bnode *temp,*tail=NULL,*head=NULL;
 int num;
 cin >> num;
 head = new bnode ;  // 为新节点动态分配内存
 if (head == NULL)
 {cout << "No memory available!";
  return NULL;
 }
 else
 {head->data = num;
  head->next = NULL;
  head->pre = NULL;
  tail = head; }
 for ( int i = 0; i < n - 1; i++)
 {cin >> num;
  temp = new bnode ;// 为新结点动态分配内存
  if (temp == NULL)
  {cout << "No memory available!";
   return head;
  }
  else
  {temp->data = num;
   temp->next = NULL;
   temp->pre = tail;
   tail->next = temp;
   tail = temp;
  }
 }
 return head ;
}
```

例子8：编写函数，将整数n插入到一个已排序的双向链表中(从小到大)

```
bnode * insertData(int n, bnode * head)
{bnode *curNode = head;  // 指向当前结点
 bnode *newNode = NULL;  // 指向新建结点
 newNode = new bnode ;
 if (newNode == NULL)
 {cout << "Not memory available!";
  return head;
 }
 newNode->data = n;
```

```
while((curNode!=NULL)
       &&(curNode->next!=NULL)
       &&(curNode->data<n))
   curNode = curNode->next;
if((curNode==NULL)||(curNode->pre==NULL))
{newNode->next = curNode;
 newNode->pre = NULL;
 if (curNode != NULL)
  curNode->pre = newNode;
 return newNode;
}
else
```

```
 {if(curNode->data>=n)
  {curNode->pre->next = newNode;
   newNode->next = curNode;
   newNode->pre = curNode->pre;
   curNode->pre = newNode;
  }
  else
  {curNode->next = newNode;
   newNode->next = NULL;
   newNode->pre = curNode;
  }
  return head;
 }
}
```

例子9：编写函数，在双向链表中查找并删除指定整数n

```
bnode *deleteData(int n,bnode *head)
{ bnode *curNode = head;
  while ( curNode && curNode->data!=n )
   curNode = curNode->next;
  if (curNode == NULL)
  {cout<<"Can't find "<< n << endl;
   return head;
  }
```

```
 if (curNode->pre == NULL)
 {head = head->next;
  head->pre = NULL;
 }
 else
 {curNode->pre->next = curNode->next;
  if (curNode->next != NULL)
   curNode->next->pre=curNode->pre;
 }
 delete curNode;
 return head;
}
```

Thank you！