C语言

第11讲：结构体

赵岩

哈尔滨工业大学软件学院

August 20, 2011

目录

目录

数据类型

- 数据类型本不存在
  - 内存里的内容，你认为它是什么，它就是什么
- 一般的CPU只支持两种类型
  - 整数、浮点数
- 高级语言设计了基本数据类型
  - 整型、浮点型、字符型等
  - 不同语言会定义不同的基本类型
  - 基本数据类型并不能方便地解决所有问题

抽象数据类型

- 用户自己构造数据类型-复合数据类型
  - 表示复杂的数据对象，典型的代表就是“结构体”，数组、指针也可算作此类
- 抽象数据类型（Abstract Data Type，简称ADT）
  - 在复合数据类型基础上增加对数据的操作
    - C++中的类——跨时代的进步
    - 例如汽车就是一种ADT

如何保存这个表

- 在程序里表示一个人（姓名、年龄、性别、.....），怎么表示？
- 想表示多个人呢？
- 如何用计算机程序实现下述表格的管理？

学号	姓名	性别	入学时间	计算机原理	英 语	数 学	音 乐
1	令狐冲	男	1999	90	83	72	82
2	林平之	男	1999	78	92	88	78
3	岳灵珊	女	1999	89	72	98	66
4	任莹莹	女	1999	78	95	87	90
5	……						
6	……						

分散数组的表示

```
1  /* 管理30个学生, 每个学生的学号用数组的下标表示*/
2  int   studentId[30];
3  char  studentName[30][10];
4  char  studentSex[30][3];
5  int   timeOfEnter[30];    /*入学时间用int表示*/
6  int    scoreComputer[30]; /*计算机原理课的成绩*/
7  int   scoreEnglish[30];  /*英语课的成绩*/
8  int   scoreMath[30];     /*数学课的成绩*/
9  int   scoreMusic[30];     /*音乐课的成绩*/
```

```
int studentId[30] = {1,2,3,4,5,6};
char studentName[30][10] = {{"令狐冲"},
                            {"林平之"},
                            {"岳灵珊"},
                            {"任莹莹"}};
char studentSex[30][3] = {{"男"},{"男"},
                          {"女"},{"女"}};
int timeOfEnter[30] = {1999,1999,1999,1999};
int scoreComputer[30] = {90,78,89,78};
int scoreEnglish[30] = {83,92,72,95};
int scoreMath[30] = {72,88,98,87};
int scoreMusic[30] = {82,78,66,90};
```

缺点

- 分配内存不集中，寻址效率不高
- 对数组进行赋初值时，容易发生错位
- **结构显得比较零散，不容易管理**

1
2
3
4
……

令狐冲
林平之
岳灵珊
任莹莹
……

男
男
女
女
……

1999
1999
1999
1999
……

90
78
89
78
……

83
92
72
95
……

72
88
98
87
……

82
78
66
90
……

希望的内存分配图

1
令狐冲
男
1999
90
83
72
82

2
林平之
男
1999
78
92
88
78

3
岳灵珊
女
1999
89
72
98
66

4
任莹莹
女
1999
78
95
87
90

结构体的解决方法

```
1  struct  STUDENT {
2    int   studentID;          /*每个学生的序号*/
3    char  studentName[10];/*每个学生的姓名*/
4    char  studentSex[4];  /*每个学生的性别*/
5    int   timeOfEnter;        /*每个学生的入学时间*/
6    int   scoreComputer;  /*计算机原理成绩*/
7    int   scoreEnglish;       /*英语成绩*/
8    int   scoreMath;      /*数学成绩*/
9    int   scoreMusic;         /*音乐成绩*/
10 };
```

struct的大小尺寸

- 一个结构变量的成员变量在内存中是相邻的
- 整个结构变量的将占用多少内存呢？
  - 是所有成员变量的内存总和吗？
  - 事实上，所有数据类型在内存中都是从偶数地址开始存放的，且结构所占的实际空间一般是按照机器字长对齐的
  - 不同的编译器、不同的平台，对齐方式会有变化，不过一般的编译器都可以设定按照多大对齐
  - 我们可以用sizeof来获得结构的大小

struct的尺寸

```
1 struct Number {
2 char c;
3 short s;
4 int i;
5 } num1;
6 printf("%d",sizeof(num1));
```

sizeof

- 它是一个C语言的关键字，并不是函数
- 可以用两种形式使用
  - sizeof(表达式) 常使用sizeof(变量名)
  - sizeof(类型)
- 求出的结果为表达式值所属类型或者类型占用的字节数，类型size_t

struct的特点

- 一个普通的类型
  - 可以定义该类型的变量、数组、指针.....
  - 它的成员可以是任意类型
    - 基本类型、数组、指针、结构...
  - 可以做函数的参数类型和返回值类型
- `struct`类型的变量
  - 可以互相赋值
  - 可以& (取地址)
  - 不可以参与运算
- 它的成员也都是如假包换的变量
- 面向对象和数据库是struct的思想的发展

结构体变量定义-1

- 先定义结构体类型再定义变量名
- 用typedef为已存在的类型定义新名字

```
1  typedef struct student
2  {
3      int      num;
4      char     name[20];
5      char     sex;
6      int      age;
7      float    score;
8      char     addr[30];
9  } STUD;
10 STUD student1,student2;
```

结构体变量定义-2

- 定义类型的同时定义变量

```
1  struct student
2  {
3      int      num;
4      char     name[20];
5      char     sex;
6      int      age;
7      float    score;
8      char     addr[30];
9  } student1,student2;
10
11 struct student student3;
```

结构体变量定义-3

- 只定义变量不定义类型

```
1  struct
2  {
3      int      num;
4      char     name[20];
5      char     sex;
6      int      age;
7      float    score;
8      char     addr[30];
9  } student1,student2;
10
11 student3; /*difficult to declare */
```

结构体变量定义思考题

```
1  struct student
2  {
3  ...;
4  };
5
6  struct student zhaoyan;     /* It works? */
7  student zhaoyan;   /* Can this work? */
8
9  typedef struct
10 {
11 ...;
12 }student;
13 student zhaoyan;   /* It works? */
```

结构体数组声明

```
1  struct  STUDENT{
2      int      studentID;
3      char     studentName[10];
4      char     studentSex[4];
5      struct   date timeOfEnter;
6      int      scoreComputer;
7      int      scoreEnglish;
8      int      scoreMath;
9      int      scoreMusic;
10 };
11 struct STUDENT students[30];
12 students[0].studentName;
13 students[0].studentSex;
```

结构体指针

```
1  struct point
2  {
3      int x;
4      int y;
5  };
6  struct point  pt;   /*定义结构体变量*/
7  struct point *ppt;  /*定义结构体指针*/
8  ppt = &pt;
9  pt.x = 0;           /*成员运算符*/
10 (*ppt).x = 0;
11 ppt->x = 0;   /*指针运算符*/
```

结构体数组和指针

```
1  struct STUDENT *pt; int i;
2  float  sum[2] = {0.0},average[2] = {0.0};
3  char *name[]={"Computer", "English"};
4  pt = students; /*pt指向结构体数组的第一个元素*/
5   for (pt=students; pt<students+30; pt++){
6    sum[0] = sum[0] + pt->scoreComputer;
7    sum[1] = sum[1] + pt->scoreEnglish;
8   }
9   for (i=0; i<2; i++){
10   average[i] = sum[i]/30;
11   printf("%20s:%4.2f\n",name[i],*(average+i));
12  }
```

结构体变量引用-成员运算符

- 结构体变量.成员名

```
1  typedef struct student
2  {
3      int      num;
4      char     name[20];
5  } STUD;
6  STUD student1,student2;
7  student1.num = 1;
8  strcpy(student2.name,"zhao yan");
```

结构体变量引用-箭头运算法

c指向结构体变量的指针->成员名

```
1  typedef struct student
2  {
3      int      num;
4      char     name[20];
5  } STUD;
6  STUD student1,pstudent;
7  pstudent = &student1;
8  pstudent->num = 1;
9  strcpy(pstudent->name,"zhao yan");
```

结构体和函数

- 向函数传递结构体的单个成员
  - 单向值传递，函数内对结构内容的修改不影响原结构
- 向函数传递结构体的完整结构
  - 单向值传递，函数内对结构内容的修改不影响原结构，**开销大**
- 向函数传递结构体的首地址
  - 用结构体数组或者结构体指针做函数参数
  - 除提高效率外，还可以修改结构体指针所指向的结构体的内容

结构体做参数

```
1       typedef struct{
2       int x;
3       int y;}point;
4       point mid, begin={1,1}, end = {0, 0};
5       point MidPt(point pt1, point pt2){
6           point pt;
7           pt.x = (pt1.x+pt2.x)/2;
8           pt.y = (pt1.y+pt2.y)/2;
9           return pt; };
10      mid = MidPt(begin, end);
```

问题

整个过程中发生了几次结构体的拷贝？

结构体做参数和返回值

- 这样呢？

```
1  void MidPt(const point* pPt1,
2          const point* pPt2, point* pMid)
3  {
4      pMid->x = (pPt1->x + pPt2->x)/2;
5      pMid->y = (pPt1->y + pPt2->y)/2;
6  }
7  MidPt(&begin, &end, &mid)
```

- 指针传递效率更高，内容传递更直观

洗牌和发牌模拟-1

```
1  char *suit[] = {"Spades","Hearts",
2                  "Clubs","Diamonds"};
3  char *face[] = {"A","2","3","4","5",
4                  "6","7","8","9","10",
5                  "jack","Queen","King"};
6  void  FillCard(struct CARD wCard[],
7          char *wFace[], char *wSuit[]) {
8      int i;
9      for (i=0; i<52; i++){
10         strcpy(wCard[i].suit, wSuit[i/13]);
11         strcpy(wCard[i].face, wFace[i%13]);
12     }
13 }
```

洗牌和发牌模拟-2

```
1  void Shuffle(struct CARD *wCard)
2  {
3      int      i,j;
4      struct card temp;
5      for (i=0; i<52; i++)
6      {
7          j = rand()%52;
8          temp      = wCard[i];
9          wCard[i] = wCard[j];
10         wCard[j] = temp;   /* 洗牌过程 */
11     }
12 }
```

洗牌和发牌模拟-3

```
1  void Deal(struct CARD *wCard)
2  {
3      int i;
4      for (i=0; i<52; i++)   /*输出发牌结果*/
5      {
6      printf("%10s%10s\n", wCard[i].suit,
7          wCard[i].face);
8      }
9  }
```

结构体的嵌套

```
1  struct  STUDENT{
2  int studentID;
3  char studentName[10];
4  char  studentSex[4];
5  struct date timeOfEnter;
6  int  scoreComputer;
7  int  scoreEnglish;
8  int  scoreMath;
9  int  scoreMusic;
10 };
11 struct STUDENT stu[30] =
12  {{1,"令狐冲","男",{1999,12,20},90,83,72,82},
13   {2,"林平之","男",{1999,07,06},78,92,88,78}};
```

结构体嵌套

```
1  struct point
2  {
3      int x; int y;
4  };
5
6  struct rect
7  {
8      struct point pt1;
9      struct point pt2;
10 };
```

结构体嵌套

- 下面的结构什么意思？

```
1 struct something{
2     struct something obj1;
3     struct something obj2;
4 };
```

```
1 struct something{
2     char name[10];
3     struct something* pOtherObj;
4 };
```

- 第一个结构无限循环定义了，第二个没问题。

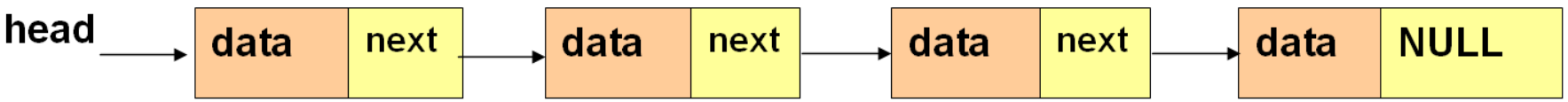
链表

- 链表：线性表的链式存储结构
  - 其特点是用一组任意的存储单元存储线性表的数据元素；存储单元可以是连续的，也可以是不连续的
  - 为了表示每个元素与后继元素的逻辑关系，除了存储元素本身的信息外，还要存储其直接后继的信息。两部分信息组成一个结点，每个结点包含：
    - 数据域：存储数据元素信息
    - 指针域：存储直接后继的存储位置信息
  - n个结点链接成一个链表（因为只包含一个指针域，故又称线性链表或单链表）

数据结构-链表

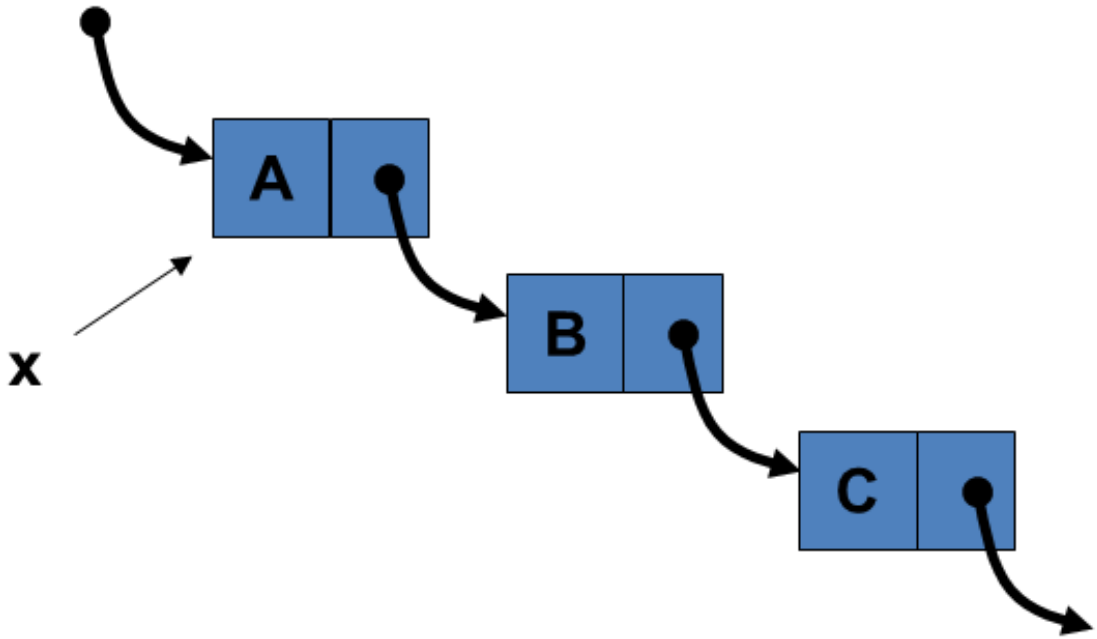
- 结构体声明时不能包含自我，但可以包含指向本结构体类型的指针变量
- 链表（Linked table）

```
1 struct node
2 {
3     int      data;
4     struct node *next;
5 }
```



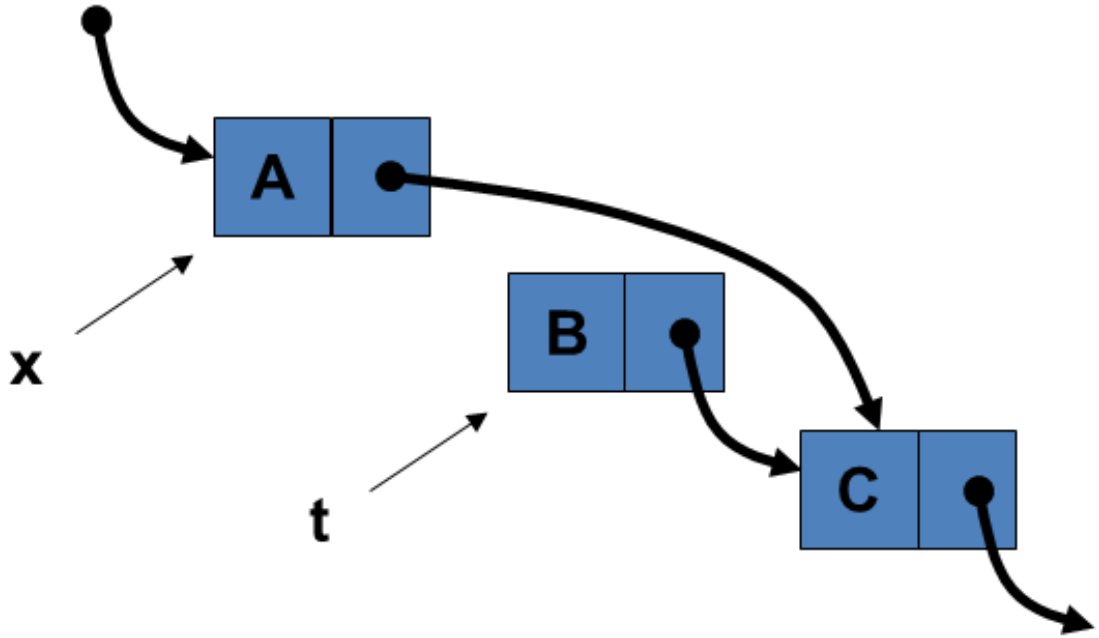
链表节点删除

```
1  t = x->next;
2  x->next = t->next;
3  free(t);
```



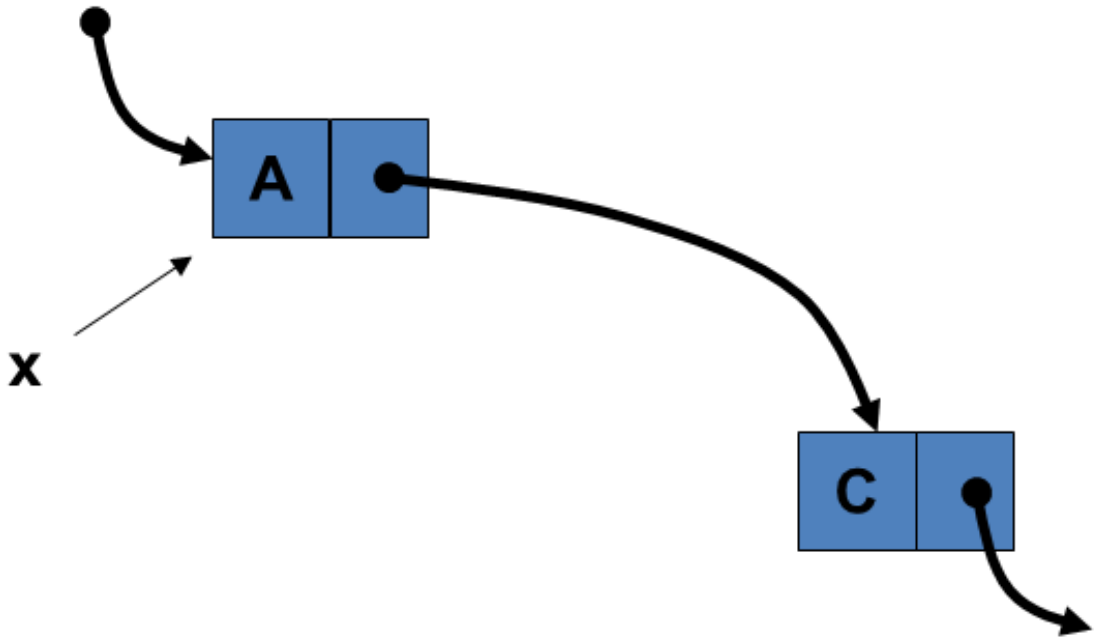
链表节点删除

```
1  t = x->next;
2  x->next = t->next;
3  free(t);
```



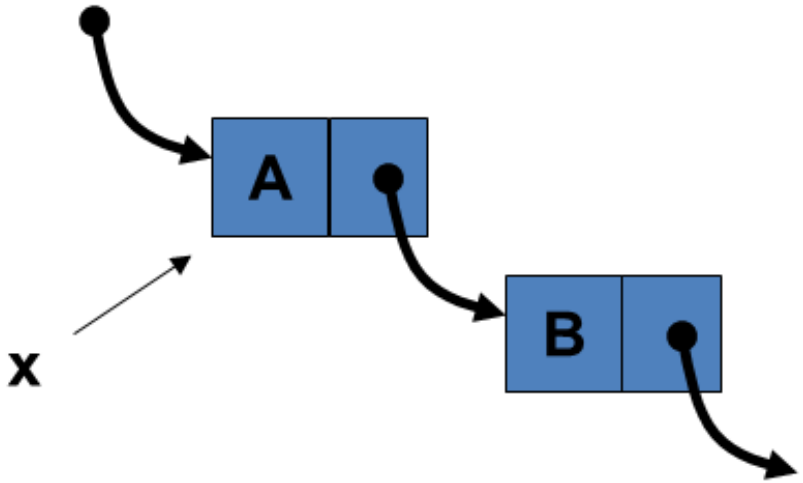
链表节点删除

```
1  t = x->next;
2  x->next = t->next;
3  free(t);
```



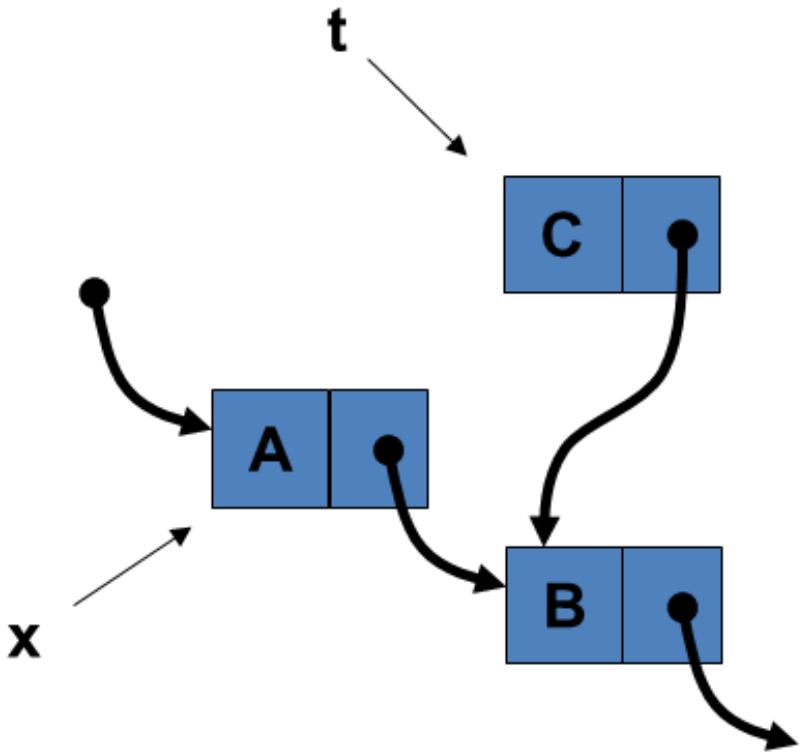
链表节点插入

```
1  t = malloc(sizeof(struct Node));
2  t->next = x->next;
3  x->next = t;
```



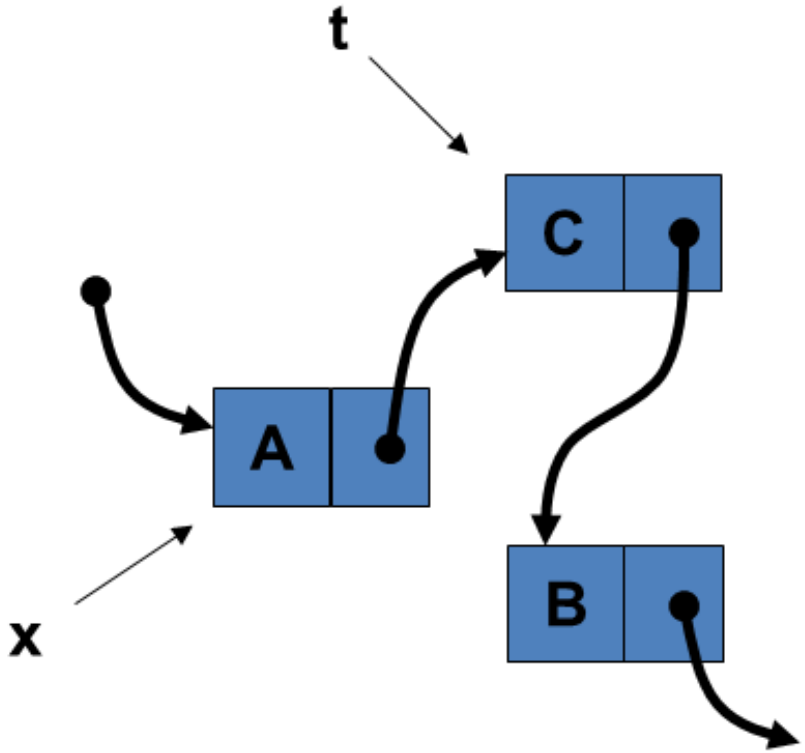
链表节点插入

```
1  t = malloc(sizeof(struct Node));
2  t->next = x->next;
3  x->next = t;
```



链表节点插入

```
1  t = malloc(sizeof(struct Node));
2  t->next = x->next;
3  x->next = t;
```

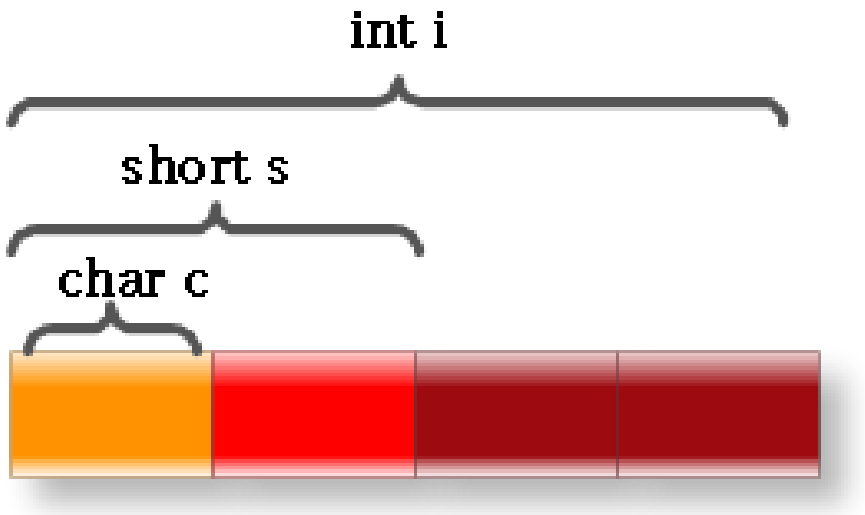


结构体和共用体

- 结构体：
  - 把关系紧密且逻辑相关的多种不同类型的变量组织到统一的名字之下，也称复合数据类型
  - 这种类型的变量占用相邻的一段内存单元
- 共用体：
  - 把**情形互斥**但又逻辑相关的多种不同类型的变量组织在一起
  - 这种类型的变量占用同一段内存单元，因此每一时刻只有一个数据起作用

共用体例子

```
1  union number{
2  char c;
3  short s;
4  int i;
5  };
```



共用体的大小

- c、s和i处于同样的地址
- sizeof(union number)
- 取决于占空间最多的那个成员变量

```
1 union number
2 {
3     char c;
4     short s;
5     int i;
6 };
7 sizeof(number);
```

共用体应用

```
1  struct person{
2      char name[20];
3      char sex;
4      int age;
5      union{
6          int single;
7          struct{
8              char spouseName[20];
9              int child;
10         }married;
11         struct date divorcedDay;
12     }marital;
13     int marryFlag;
14 };
```

共用体应用改进

```
1  struct person{
2      char name[20];
3      char sex;
4      int age;
5      union{
6          int houseNumber; /*房子的数目*/
7          struct{
8              char spouseName[20];
9              int child;
10         }married;
11         struct date divorcedDay;
12     }marital;
13     int marryFlag;
14 };
```

共用体总结

- **情形互斥下，节省内存**
- 同一内存单元在每一瞬时只能存放其中一种类型的成员；并非同时都起作用
- 起作用的成员是最后一次存放的成员

位段

- 想表达人的姓名、出生年、月、日，都定义什么类型的成员变量？
- 这样有很多的空间浪费
  - 比如month只可能取值1-12，4bits足够

```
1 struct person
2 {
3     char name[12];
4     int  year;
5     char month;
6     char day;
7 };
```

位段的定义

- 调整成员顺序可以让结构更紧凑
- 每个位段都可以当作一个无符号整型数使用
- 表达范围受限

```
1 struct person{
2     char name[12];
3     unsigned int  year  : 12;
4     unsigned int   month : 4
5     unsigned int  day   : 5;
6 };
```

总结


结构体
共用体
结构体
结构体声明和定义
结构体和函数
结构体嵌套　　链表
结构体的成员访问
共用体
使用上互斥
只有一个变量有效

谢谢大家，欢迎提问！