目录

[结构体复习 1](#_Toc27488919)

[指向结构体的指针 2](#_Toc27488920)

[基础 2](#_Toc27488921)

[改进 3](#_Toc27488922)

[链表 3](#_Toc27488923)

[思考1： 4](#_Toc27488924)

[思考2：可以不用变量过渡 5](#_Toc27488925)

[思考3： 5](#_Toc27488926)

[思考4： 6](#_Toc27488927)

[思考5：以下错误程序，前述已指明 6](#_Toc27488928)

[文件 7](#_Toc27488929)

[1、思考1： 7](#_Toc27488930)

[2.思考2： 7](#_Toc27488931)

[3、思考3： 8](#_Toc27488932)

[思考4： 9](#_Toc27488933)

## 结构体复习

#include<stdio.h>

struct student

{ char name[20];

int age;

};

int main()

{ struct student stu1,stu2;

scanf("%s%d",stu1.name,&stu1.age);

printf("姓名是:%s,年龄是%d",stu1.name,stu1.age);

return 0;

}

## 指向结构体的指针

### 基础

#include<stdio.h>

struct student

{ char name[20];

int age;

};

int main()

{ struct student stu1,**\*p**;

//int x,\*p//p=&x;

p=&stu1;

scanf("%s%d",**(\*p).name,&((\*p).age))**;

printf("姓名是:%s,年龄是%d",stu1.name,stu1.age);

return 0;

}

### 改进

#include<stdio.h>

struct student

{ char name[20];

int age;

};

int main()

{ struct student stu1,**\*p**;

p=&stu1;//**(\*p).name<====>p->name**

scanf("%s%d",p->name,&p->age);

printf("姓名是:%s,年龄是%d\n",stu1.name,stu1.age);

printf("姓名是:%s,年龄是%d\n",p->name,p->age);

return 0;

}

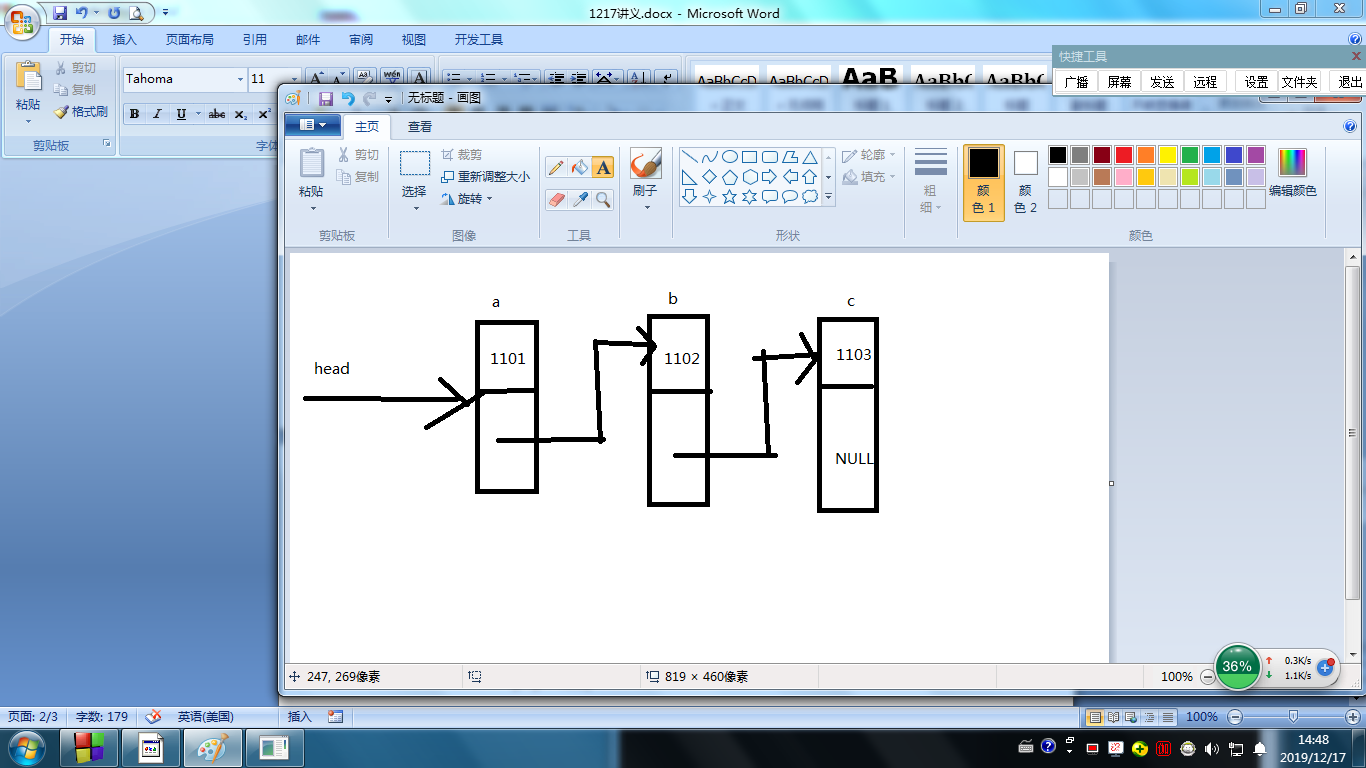
## 链表

#include<stdio.h>

struct student

{ int num;

struct student \*next;

}\*head,\*p,a,b,c;

int main()

{  **//链表结点的初始化**

a.num=1101;b.num=1102;c.num=1103;

a.next=&b;b.next=&c,c.next=NULL;//建立链的关系

p=&a;//将p指针指向第一个

//遍历链表

while(p!=NULL)

{

printf("%d-->",p->num);

p=p->next;//指向下一个

}

printf("\n");

p=&a;

a.next=&c;**//等同于将b结点跳过（或称为删除）**

while(p!=NULL)

{

printf("%d-->",p->num);

p=p->next;

}

return 0;

}

### 思考1：

#include<stdio.h>

struct student

{ int num;

struct student \*next;

}\*head,\*p;

int main()

{ p->num=1101;//(\*p).num<===>p->num;/**/此句有错**

printf("%d",p->num);

return 0;

}

上述错误的原因，等同如下错误

int \*p;

\*p=3;//因为指针地址没有初始化

这处改进：int x,\*p;//**必须借助x来过渡**

p=&x,//初始化一个变量地址给它

\*p=3;//赋值

### 思考2：可以不用变量过渡

#include<stdio.h>

#include<stdlib.h>

int main()

{ int \*p,x;

float y=4.123;

x=(int)y;//此句功能是将y转换成整数，再给x

p=(int \*)malloc(sizeof(int));

//malloc功能，在内存闲置空间中，申请指定大小的空间

\*p=3;

printf("%d\n",\*p);

printf("%d,%d",sizeof(int),sizeof(char));//4,1

}

### 思考3：

#include<stdio.h>

struct student

{ int num;

struct student \*next;

}\*p;

int main()

{

p=(struct student \*) malloc(sizeof(struct student));

p->num=1101;

p->next=NULL;

printf("%d",p->num);

return 0;

}

### 思考4：

#include<stdio.h>

struct student

{ int num;

struct student \*next;

}\*p,a;

int main()

{

p=&a;//借助临时的a结构体变量，就避开了动态内容分配，当然，此句必须有

p->num=1101;

p->next=NULL;

printf("%d",p->num);

return 0;

}

### 思考5：以下错误程序，前述已指明

#include<stdio.h>

struct student

{ int num;

struct student \*next;

}\*p,a;

int main()

{

// p=&a;//借助临时的a结构体变量，就避开了动态内容分配，当然，此句必须有

p->num=1101;

p->next=NULL;

printf("%d",p->num);

return 0;

}

## 文件

### 1、思考1：

#include<stdio.h>

int main()

{ FILE \*fp;//定义一个指向结构体的文件指针

fp=fopen("d:\\c1217\\kk.txt","w");

//也可以 fp=fopen("d:/c1217/kk.txt","w");

fclose(fp);

return 0;

}

### 2.思考2：

#include<stdio.h>

int main()

{ FILE \*fp;//定义一个指向结构体的文件指针

fp=fopen("test.txt","w");

//不加路径，test.txt存在于当前.c的同一位置

fclose(fp);

return 0;

}

### 3、思考3：

#include<stdio.h>

int main()

{ FILE \*fp;//定义一个指向结构体的文件指针

fp=fopen("test.txt","w");

//w参数，就是新建文件，若存在，则覆盖（删除）

//不加路径，test.txt存在于当前.c的同一位置

if (fp==NULL)

{

printf("磁盘或文件坏，无法操作");

exit(0);//功能结束main执行。比break结束循环还要强大，类似于return 0;

}

printf("正常结束");

fclose(fp);

return 0;

}

### 思考4：

#include<stdio.h>

int main()

{ FILE \*fp;//定义一个指向结构体的文件指针

fp=fopen("test.txt",**"r")**;

//r参数，就是打开已经存在的文件（必须在）

//不加路径，test.txt存在于当前.c的同一位置

if (fp==NULL)

{

printf("磁盘或文件坏，无法操作");

exit(0);//功能结束main执行。比break结束循环还要强大，类似于return 0;

}

printf("正常结束");

fclose(fp);

return 0;

}

若提前将指定位置的目标文件，test.txt删除，则输出为“磁盘或文件坏，无法操作”，不会输出“正常结束”