C结构体和链表

追風 (/u/e1e11173f41b) + 关注
2016.12.20 21:35 字数 560 阅读 27 评论 0 喜欢 1
(/u/e1e11173f41b)

一，结构体变量定义及初始化

```
#include<stdio.h>
#include<string.h>
//定义结构体，可以在主函数内定义
struct student
{
    char name[32];
    char gender;
    int age;
//其boy是先定义到后边初始化，people是定义的同时初始化，但是这两种用法不建议使用
}boy,people = {"yyy", 'y', '0'};
int main()
{
    printf("%s\n%c\n%c\n", people.name, people.age, people.gender);
    struct student person = {"xxx", 'x', 'x'};
    printf("%s\n%c\n%c\n", person.name, person.age, person.gender);
    //对boy进行初始化
    strcpy(boy.name, "Jack");
    boy.age = 24;
    boy.gender = 'm';
    printf("%s\n%d\n%c\n", boy.name, boy.age, boy.gender);
    //对girl和girl1进行定义及初始化
    struct student girl,girl1;
    strcpy(girl.name, "Alisa");
    girl.age = 24;
    girl.gender = 'w';
    printf("%s\n%c\n%c\n", girl.name, girl.age, girl.gender);
    //结构体变量可以直接赋值给相同类型结构体
    girl1 = girl;
    printf("%s\n%c\n%c\n", girl1.name, girl1.age, girl1.gender);
    return 0;
}
```

二，无名结构体

```
#include<stdio.h>
#include<string.h>
struct
{
   char gender;
    int age;
//无名结构变量必须在定义后定义，
}stu;
int main()
{
    stu.age = 28;
    printf("%d\n", stu.age);
    return 0;
}
```

备注：无名结构体很少使用

三，宏定义结构体

```
#include<stdio.h>
#include<string.h>
struct student
{
   char gender;
    int age;
};                          //结构体的“;”不能少
#define STU struct student  //宏定义结构体
int main()
{
    STU stu, stu1;          //宏定义STU使用
    stu.gender = 'm';
    stu1.age = 25;
    printf("%d\n%c\n", stu1.age, stu.gender);
}
```

四，结构体嵌套

```
#include<stdio.h>
#include<string.h>
struct date
{
    int year;
    int month;
    int day;
};
struct student
{
    char name[32];
    int age;
    struct date birth;   //结构体嵌套
};
int main()
{
    struct student stu;
    strcpy(stu.name, jack);
    stu.age = 25;
    stu.birth.year = 1991;
    stu.birth.month = 3;
    stu.birth.day = 21;
    printf("His name is %s\nHe births in %d.%d.%d\n", stu.name,
               stu.age, stu.birth.year, stu.birth.month, stu.birth.day);
    return 0;
}
```

五，结构体数组

```
#include<stdio.h>
#include<string.h>
struct student
{
    int age;
    char name[32];
}
int main()
{
    struct student arr[3] =
{
{24, "Jack"},
{23, "Alisa"},
{25, "Jays"}
}                            //结构体数组
    printf("arr[0].age:%d\narr[1].name:%s\n",arr[0].age, arr[1].name);
    reurn 0;
}
```

六，结构体指针

```
#include<stdio.h>
#include<string.h>
struct date
{
    int year;
    int month;
    int day;
};
struct student
{
    char name[32];
    int age;
    struct date birth;
};
int main()
{
    struct student stu;
    struct student *p = &stu;      //结构体指针
    strcpy(p->name, jack);
    p->age = 25;
    p->birth.year = 1991;
    p->birth.month = 3;
    p->birth.day = 21;
    printf("His name is %s\nHe births in %d.%d.%d\n", p->name,
               p->age, p->birth.year, p->birth.month, p->birth.day);
    return 0;
}
```

七，typedef 重命名

```
#include<stdio.h>
#include<string.h>
typedef struct student
{
    int age;
    char name[32];
//相当将struct student重命名为STU，而非宏定义
}STU;
    typedef int M;
    M a = 4;      //M和int的作用是一样的，只是重命名
    int b = 5;
    M c = a + b;
    printf("%d + %d = %d\n", a, b, c);
    return 0;
}
```

八，结构体大小

```
#include<stdio.h>
struct A //8
{
    char a;
    char b;
    int c;
};
struct B  //32
{
    char a;
    int b;
    char c[23];
};
struct C  //16
{
    char a;
    int c[3];
};
struct D //6
{
    char a;
    short b;
    char c
};
struct E //4
{
    char a;
    char b;
    short c;
};
int main()
{
    printf("sizeof(struct A) = %ld\n",sizeof(struct A));
    printf("sizeof(struct B) = %ld\n",sizeof(struct B));
    printf("sizeof(struct C) = %ld\n",sizeof(struct C));
    printf("sizeof(struct D) = %ld\n",sizeof(struct D));
    printf("sizeof(struct E) = %ld\n",sizeof(struct E));
    return 0;
}
```

内存对齐:
Linux:4字节
Windows:8字节

默认从偏移量为0的位置开始存储
每个字母所占字节数是其自生大小的整数倍

九，联合体

```
#include<stdio.h>
union untype
{
    int a;
    float b;
    char c;
};
int main()
{
    union untype un;
    un.a = 123;
    un.b = 3.14;
    un.c = 'f';
    printf("%d\n", un.a);  //一次只能访问一个值
    //printf("%d\n", un.b);
    //printf("%d\n", un.c);
    return 0;
}
```

十，枚举类型

```
#include<stdio.h>
enum type
{
    A,                //若不赋值，默认为0，B为1，C为2
    B,
    C,
    D = 12,
    E = 3,
    F                 //最后一个没有“,”，F递增为4
};
int main()
{
    enum type num;
    num.A   //ERROR
    num = A;
    printf("input num:");
    scanf("%d", &num);
swich(num)
{
    case A:
        printf("0\n");
        break;
    case B:
        printf("1\n");
        break;
    default:
        break;
}
    return 0;
}
```

十一，链表

- 链式存储结构，线性存储结构，其大小可动态改变，链表由一个个结点串起来的数据链，节点由数据域和指针域。
- 分配空间

  ```
  //申请一块堆空间，大小为sizeof(struct student)

  pa = (struct date *)malloc (sizeof(struct student));

  free(pa);  //释放空间
  ```

- 创建一个头结点：
  struct student *head;*
  head = (struct student)malloc(sizeof(struct student));
  头结点标识一个链表，即链表名称
  头结点的数据域不存放数据，指针域存放下一个结点的地址，头结点只是为了标识这个链表

```
#include<stdio.h>
#include<stdlib.h>
struct student
{
  int ID;
  char name[32];
  struct student *next;
};
#define LEN sizeof(struct student)
struct student *add_link(struct student *head)
{
  struct student *temp = (struct student*)malloc(LEN);
  printf("input ID:\n");
  scanf("%d", &temp->ID);
  printf(" input name:\n");
  scanf("%s", temp->name);
  temp->next = NULL;  //可以省略
  temp->next = head->next;
  head->next = temp;
  temp = NULL; //要置空，防止成为野指针
  return head;
}
struct student *delete_head(struct student *head)
{
  struct student *temp = head->next;
  head->next = temp->next;
  free(temp);
  temp = NULL;
  return head;
}
void show_link(struct student *head)
{
  struct student *p = head->next;
  printf("ID\tname\n");
  while(p != NULL)
  {
      printf("%d\t%s\n", p->ID, p->name);
      p = p->next;
  }
}
int main()
{
//创建链表
  struct student *head;
  head = (struct student *)malloc(LEN);
  head->next = NULL;
  int i = 0;
  for(i = 0; i < 5; i++)
  head = add_link(head);    //头插插入链表调用
  show_link(head);
  head = delete_head(head); //头删删除链表调用
  show_link(head);
  return 0;
}
```

结论

1，链表是一个难点，链表的插入和删除是重点。

2，弄清链表的实现方式，并自己写出代码实现。

更多分享

nshu.io/notes/images/7796602

(/apps/download?utm_source=nbc)

登录 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-comment-form) 后发表评论

评论

智慧如你，不想发表一点想法 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-nocomments-text)咩~