

面試中常見的C語言與C++區別的問題

C語言Plus 今天

以下文章來源於C語言與CPP編程，作者自成一派123



C語言與CPP編程

分享C語言/C++，數據結構與算法，計算機基礎，操作系統等



C和C++的區別

- C語言是一種結構化語言，其偏重於數據結構和算法，屬於過程性語言
C++是面向對象的編程語言，其偏重於構造對像模型，並讓這個模型能夠契合與之對應的問題。其本質區別是解決問題的思想方法不同
- 雖然在語法上C++完全兼容C語言，但是兩者還是有很多不同之處。下面將詳細講解C和C++不同之處的常見考題

關鍵字static在C和C++區別

C和C++中都有關鍵字static關鍵字，那麼static關鍵字在C和C++中的使用有什麼區別？請簡述之。

分析問題：在C中，用static修飾的變量或函數，主要用來說明這個變量或函數只能在本文件代碼塊中訪問，而文件外部的代碼無權訪問。並且static修飾的變量存放在段存儲區。主要有以下兩種用途。

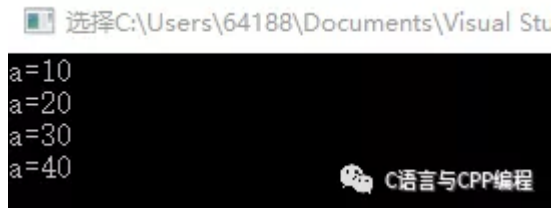
1. 定義局部靜態變量

- 局部靜態變量存儲在靜態存儲區，在程序運行期間都不會釋放，只在聲明時進行初始化，而且只能初始化一次，如果沒有初始化，其自動初始化為0或空字符。具有局部變量的“記憶性”和生存週期“全局性”特點。
- 局部變量的“記憶性”是指在兩次函數調用時，第二次調用開始時，變量能夠保持上一次調用結束數的值。如下例：

```
#include <stdio.h>

//20200505 公众号：C语言与CPP编程
void staticShow()
{
    static int a=10;
    printf("a=%d\n",a);
    a += 10;
}

int main()
{
    for(int i=0;i<4;i++)
    {
        staticShow();
    }
    return 0;
}
```



```
选择C:\Users\64188\Documents\Visual Stu
a=10
a=20
a=30
a=40
C语言与CPP编程
```

運行結果

利用生存週期的“全局性”，可以改善函數返回指針的問題，局部變量的問題在於當函數退出時其生存週期結束。而利用static修飾的局部變量卻可以延長其生存期。如下所示：

```
#include <stdio.h>
#include <string.h>

//20200505 公众号：C语言与CPP编程
char *p = NULL;
char *helloToStr(char *b)
{
    static char a[50];
    a[0]='H';
    a[1]='E';
    a[2]='L';
    a[3]='L';
    a[4]='O';
    strcpy(a+5,b);
    p=a;
    return a;
};
int main(void)
{
    printf("%s\n",helloToStr("yang"));

    strcpy(p+5,"song");
    printf("%s\n",p);
}
```

```
strcpy(p+5, "zhang");  
printf("%s\n", p);  
  
strcpy(p+5, "wang");  
printf("%s\n", p);  
  
return 0;  
}
```

C:\Users\64188\Documents\Visual Studio 2012\Projec

```
HELLOyang  
HELLOsong  
HELLOzhang  
HELLOWang
```

C语言与CPP编程

運行結果

2. 限定訪問區域

被static修飾的變量、函數只能被同一文件內的代碼段訪問。在此static不再表示存儲方式，而是限定作用範圍。如下所示：

```
//Test1.cpp  
static int a;  
int b;  
extern void fun1()  
{  
    .....  
}  
  
static void fun1()
```

```
{
    .....
}

//Test2.cpp
extern int a;          //错误·a是static类型·无法在Test2.cpp文件中使用
extern int b;          //使用Test1.cpp中定义的全局变量
extern void fun1();     //使用Test1.cpp中定义的函数
extern void fun2();     //错误·无法使用Test1.cpp文件中static函数
```

在C++中除了上述的两种常用方法外还有另外一种使用方法：定义静态成员变量和静态成员函数。静态成员变量或静态成员函数表示其不属于任何一个类实例，是类的所有类实例所共有的。如下所示：

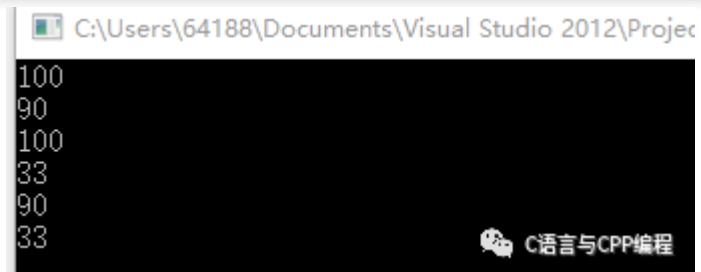
```
#include <iostream.h>
#include <string.h>
class A
{
public:
    static int a;
    static int geta();
    int b;
    int getb();
};
int A::a=100;
int A::geta()
{
    return a;
}
int A::getb()
{
    return b;
}
```

```
int main(void)
{
    A m,n;
    m.b=90;

    cout<<m.geta()<<endl;
    cout<<m.getb()<<endl;
    cout<<m.a<<endl;

    n.a=33;
    n.b=44;

    cout<<m.geta()<<endl;
    cout<<m.getb()<<endl;
    cout<<m.a<<endl;
    return 0;
}
```



```
C:\Users\64188\Documents\Visual Studio 2012\Project
100
90
100
33
90
33
C语言与CPP编程
```

运行结果

答案

在C中static用来修饰局部静态变量和外部静态变量、函数。而C++中除了上述功能外，还用来定义类的成员变量和函数，即静态成员和静态成员函数。

注意：编程时static的记忆性和全局性的特点可以使在不同时期调用的函数进行通信，传递信息，而C++的静态成员则可以在多个对象实例间进行通信，传递信息。

结构体在C语言和C++的区别

分析问题：在C中，结构体是一种简单的复合型数据，由若干个基本类型数据或复合类型数据组合而成。而在C++结构体中，还可以声明函数。如下所示：

```
#include <iostream.h>

struct A
{
public:
    int a;
    int gata()
    {
        return a;
    }
};

int main(void)
{
    m.a=50;
    cout<<m.gata()<<endl;
    return 0;
}
```

输出结果：50

但是这种用法看起来有点不伦不类，这是C到C++过渡的遗留问题

答案

- C语言的结构体是不能有函数成员的，而C++的类可以有。
- C语言结构体中数据成员是没有`private`、`public`和`protected`访问限定的。而C++的类的成员有这些访问限定（在C++中结构体的成员也是有访问权限设定的，但是类成员的默认访问属性是`private`，而结构体的默认访问属性是`public`）。
- C语言的结构体是没有继承关系的，而C++的类却有丰富的继承关系。

说明：虽然C的结构体和C++的类有很大的相似度，但是类是实现面向对象的基础。而结构体只可以简单地理解为类的前身。

C中malloc和C++的new区别

分析问题：`malloc`、`free`与`new`、`delete`都是用来动态申请内存和释放内存的。不同点如下：

- `malloc`、`free`是标准库函数，`new`、`delete`则是运算符。`malloc`、`free`在C、C++中都可使用，而`new`、`delete`只属于C++。
- `malloc`要指定申请内存的大小，其申请的只是一段内存空间。而`new`不必指定申请内存的大小，建立的是一个对象。
- `new`、`delete`在申请非内部数据类型的对象时，对象在创建的同时会自动执行构造函数，在消亡时会自动执行析构函数，这不在编译器的控制之内，所以`malloc`、`free`无法实现。
- `new`返回的是某种数据类型的指针，而`malloc`返回的是`void`型指针。

- 由于new、delete是运算符，可以重载，不需要头文件的支持，而malloc、free是库函数，可以覆盖，并且要包含相应的头文件。

答案

1. new、delete是操作符，可以重载，只能在C++中使用。
2. malloc、free是函数，可以覆盖，C、C++中都可以使用。
3. new可以调用对象的构造函数，对应的delete调用相应的析构函数。
4. malloc仅仅分配内存，free仅仅回收内存，并不执行构造和析构函数。
5. new、delete返回的是某种数据类型指针，malloc、free返回的是void指针。

注意：malloc申请的内存空间要用free释放，而new申请的内存空间要用delete释放，不能混用。因为两者实现的机理不同。

C++引用和C的指针有何区别

分析问题：引用就是变量或对象的别名，它不是值，不占据存储空间，其只有声明没有定义。在C++中引用主要用于函数的形参和函数返回值。

1、作为函数的参数

当函数的返回值多于一个时，可以使用指针实现。如下所示：

```
void swap(int *a, int *b)
{
    int temp;
```

```
    temp = *a;
    *a = *b;
    *b = temp;
}

int main(void)
{
    int a=10,b=5;
    cout<<"Before change:"<< a<<" "<<b<<endl;
    swap(&a,&b);
    cout<<"After change:"<< a<<" "<<b<<endl;
    return 0;
}
```

输出结果：

Before change: 10 55

After change: 55 10

虽然上述代码实现了多个返回值的功能，但是函数的语法相对传值方式比较麻烦。在函数中使用指针所指对象的数值时，必须在指针前加上*，如上例中的的swap函数频繁使用了“*a”、“*b”，如此不仅书写麻烦，还不利于阅读，并且容易产生错误。在函数调用时也容易产生误解，如上述代码main函数中swap(&a, &b)，看起来好像是交换了两个变量的地址似的。而用引用实现swap函数，如下所示：

```
void swap(int &a, int &b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}

int main(void)
```

```
{  
    int a=10,b=5;  
    cout<<"Before change:"<< a<<" "<<b<<endl;  
    swap(a,b);  
    cout<<"After change:"<< a<<" "<<b<<endl;  
    return 0;  
}
```

可以看出用引用实现swap函数比指针实现要简洁，调用也显得更加合乎情理。

2、引用作为函数的返回值

在大多数情况下可以被指针替代，但是遇到构造函数和操作符重载函数的“形式自然”的问题时，是不能被指针替代的。指针和引用功能相似，但是在操作时却有很多不同的地方，如指针的操作符是“*”和“->”，而引用常用的操作符是“.”。在使用时还要注意以下几点：

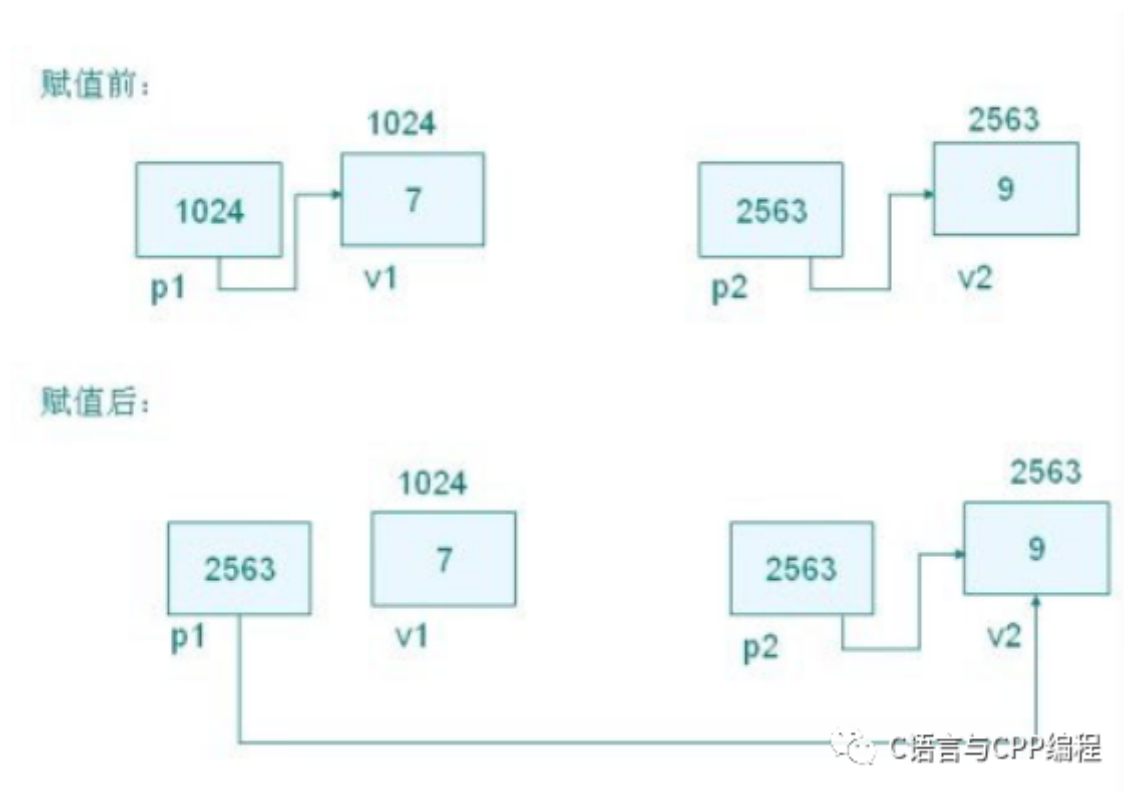
1. 指针可不初始化且初始化的时候，可以指向一个地址，也可以为空。引用必须初始化且只能初始化为另一个变量，如下：

```
int a=1024;  
int *p=&a;  
int &b=a;
```

2. 引用之间的赋值和指针之间的赋值不同。指针赋值如下：

```
int a=1,b=2;  
int *p1=&a, *p2=&b;
```

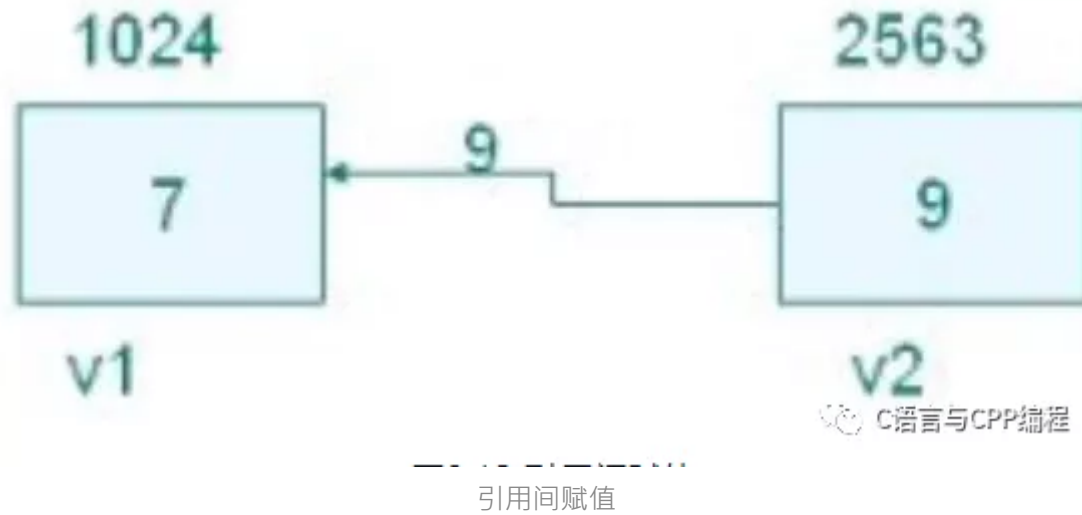
这时执行`p1=p2`；后，`p1`原来指向的对象`v1`的值并没有改变，而`p1`被赋值为`p2`所指向的对象，如下图：



引用赋值如下：

```
int a=1,b=2;
int &v1=a, &v2=b;
```

这时执行`r1= r2`；改变的是`v1`，将`r 2`指向的对象的值赋值给`v1`，而不是引用`r1`本身。赋值之后，两个引用还是指向各自的原来对象，如图下图。



3. 指针可以被重新赋值以指向另一个不同的对象。但是引用则总是指向在初始化时被指定的对象，以后不能改变。如下所示：

```
int a=1;
int b=2;
int &v1=a;    //v1引用a
string *pi = &a; //pi指向a
v1=b;        //v1仍旧引用a，但是a现在的值是2；
pi=&b;        //pi指向b，a没有改变
```

答案

指针和引用主要有以下区别：

- 引用必须被初始化，但是不分配存储空间。指针不声明时初始化，在初始化的时候需要分配存储空间。

- 引用初始化以后不能被改变，指针可以改变所指的对象。
- 不存在指向空值的引用，但是存在指向空值的指针。

注意：引用作为函数参数时，会引发一定的问题，因为让引用作为参数，目的就是想改变这个引用所指向地址的内容，而函数调用时传入的是实参，看不出函数的参数是正常变量，还是引用，因此可能会引发错误。所以使用时一定要小心谨慎。



C语言Plus

C/C++开发、最新资讯，打造C/C++程序员最喜爱的交流平台！吐槽、吹水、开车，我们无“恶”不做！学习进阶、分享经验、招聘内推我们享...
150篇原创内容



公众号

喜欢此内容的人还喜欢

頂級C程序員之路

極客重生

字符串之正則表達式

C語言與CPP編程

比特幣又爆倉了.....

編程指北

(0.45 seconds)

50 條

编程语言	1
C	1
Python	1
Java	1
C++	2



