# 常见笔试题

1. 以下程序的输出结果是什么？\_\_A\_\_
2. #include<stdio.h>
3. int main()
4. {
5. int arr[]={11,12,13,14,15};
6. int \*ptr = arr;
7. \*(ptr++) += 100; //地址自加，ptr 指向空间改变
8. printf("%d %d\n",\*ptr,\*(++ptr));//stdcall 🡪 标准调用
9. return 0;
10. }

A:13 13 B:112 13

C:12 12 D:12 13

2. 以下代码说法正确的是: C

1. #include<stdio.h>
2. int main(void)
3. {
4. char aChar;
5. int aInt;
6. aInt = aChar = -120;
7. printf("%d\n",aInt);
8. return 0;
9. }

A:一定输出-120 B:一定不能输出-120

C:可能输出-120 D:输出%d

二、填实空题：

1、typedef union {long i; int k[5]; char c;} DATE;   
struct data { int cat; DATE cow; double dog;} too; --32  
DATE max; --20  
则语句printf("%d",sizeof(struct date)+sizeof(max)); 的  
执行结果是：\_\_\_52\_\_

1. 简答题：
2. **用变量a 给出下面的定义**  
   e) 一个有 10个指针 的数组，该指针是指向一个整型数的;

int\* a[10]={……..};  
f)  一个指向有10个整型数数组的指针;

int (\*a)[10]={…};  
g) 一个指向函数的指针，该函数有一个整型参数并返回一个整型数:

int (\*a)(int) = …;  
h) 一个有10个指针的数组，该指针指向一个函数，该函数有一个整型参数并返回一个整型数;

int (\*a[10]) (int)={…};

1. 以下代码有什么问题

int main()  
{  
char a;  
char \*str = &a;  
strcpy(str,"hello");   
printf(str);  
return 0;  
}

str 指向的空间 是 char 类型 只有一个字节的大小

hello –》空间大小是6，越界

段错误

将一个字符串复制进一个字符变量指针所指地址，虽然可以正常输出,但因为越界进行正常读写而导致程序崩溃

1. 请问以下代码有什么问题：  
   char\* s="AAA";  
   printf("%s",s);  
   s[0]='B';  
   printf("%s",s);

:字符串 是常量空间 --》 不能被修改

1. c和c++ 中的struct有什么不同？

C的struct ：是结构体（复杂类型）,只能定义变量

C++ 的 struct 是 类 ： 既能定义变量，也可定义方法

C++的struct和class区别:主要区别，默认存储权限不同

struct : 默认权限public

Class : 默认权限 private

1. 会出现什么问题？

void getmemory(char \* p) //改成 char\* &p  
{  
p=(char \*) malloc(100);  
strcpy(p,“hello world”); 🡪没有释放 –} 内存泄漏  
}  
int main( )  
{  
char \*str=NULL;  
getmemory(**str**);  
printf(“%s/n”,str);

free(str); //访问非法空间地址  
  
return 0;  
}

值传递

str 只是将 NULL值 传递给 p ,str并没有被改变， free(str)，就是free(NULL),会出现问题

程序崩溃，并不能传递动态内存，（动态如何理解? 程序员申请，释放，可动）

1. 下列代码出现问题

void main()  
{  
char aa[10];  
printf(“%d”,strlen(aa));  
}

//aa 没有被初始化，strlen 获取字符串长度，数组里面没有长度，它是以’\0’来判断字符串长度，但aa没有被初始化，也就是说，它的数据时不确定的，‘\0’可能出现在任何位置

1. 要对绝对地址0x100000赋值，我们可以用(unsigned int\*)0x100000 = 1234;那么要是想让程序跳转到绝对地址是0x100000去执行，应该怎么做？

强转成函数指针 然后调用执行

Typedef Void(\* pfun)()

(\*(pfun)0x1000000) ();

1. 关于内存的思考题（1）你能看出有什么问题？

**void GetMemory(char \*p)**  
**{**  
**p = (char \*)malloc(100);**  
**}**  
**void Test(void)**  
**{**  
**char \*str = NULL;**  
**GetMemory(str);**  
**strcpy(str, "hello world");**  
**printf(str);**  
**}**

**1.str未发生改变,依然是NULL,**

**2.程序执行到strcpy 时，因为 str 并没有空间,会发生错误**

**3.申请空间 ， 并没有释放**

9、#include  <filename.h>    和#include “filename.h” 有什么区别？

<filename.h>:在系统默认路径下 查找,

“filename.h”:在本文件路径下查找，找不到，在在系统路径下查找

10、const   有什么用途？（请至少说明两种）

1.const : 常变量,初始化后就不能被修改（必须初始化）

2.const可以修饰函数的参数、返回值，甚至函数的定义体。被const 修饰的东西都受到强制保护，可以预防意外的变动，能提高程序的健壮性。

3.在类中，可以用作 常成员变量，常成员函数（起到限定作用，在该函数内不能修改类中的成员变量，但可以引用）

11、static有什么用途？（请至少说明两种）

1.static 修饰的变量 在静态区,程序运行时分配空间，程序结束时释放空间

2.static修饰的只能在本文件访问，外文件不能访问

3.在类中:作为静态成员,被static修饰的成员 称为类成员，这个成员是属于这个类的，所有该类型对象公有

生命周期 和 作用域：

1. 在函数体，一个声明为静态的变量,在这一函数被调用过程中，维持其值不变
2. 在模块（文件）内（但在函数体外）,一个声明为静态的变量可以被模块内所有函数访问,但不能被模块外其他函数访问，它是一个本地全局变量 –》 作用域
3. 在模块内,一个被声明为静态的函数只可被这一模块内其他函数调用,那就是被限制在声明它的模块的 本地范围 内使用

12、队列和栈有什么区别？ （尝试默写泛型栈）

1、队列:FIFO(先进先出)，栈：FILO（先进后出）

2、对插入和删除不同: 队列对尾进，队头删，两端 ，栈：栈顶进栈顶删，一端

3、遍历数据速度不同

//栈

#define MAX 10

Template<class T>

Class stack{

Public:

Stack():top(-1){}

//栈满

Bool empty();

//栈空

Bool full();

//栈的长度

Unsigned int size();

//入栈

Void push(T);

//出栈

Void pop();

//取栈

T top();

Private:

T buf[MAX];//容器长度

int head;//栈顶

};

Template<class T>

Bool stack<T>::empty()

{

If(this->head==-1)

Return true;

Return false;

}

Template<class T>

Bool stack<T>::full()

{

If(this->head+1==MAX)

Return true;

Return false;

}

Template<class T>

Unsigned int stack<T>::size()

{

Return this->head+1;

}

Template<class T>

Void stack<T>::push(T data)

{

If(this->full())

Return ;

This->buf[++this->head]=data;

}

Template<class T>

Void stack<T>::pop()

{

If(this->empty())

Return;

this->head--;

}

Template<class T>

T stack<T>::top()

{

Return this->buf[this->head];

}

13、**Heap（堆）与stack（栈）的差别。**

1.heap是堆，stack是栈

2.stack的空间由操作系统自动分配和释放，heap的空间是手动申请和释放的

3.stack空间有限，heap的空间是很大的自由区。

**14、用宏定义写出swap（x，y），即交换两数。**

**SWAP(x,y) (x)+=(y);\**

**(y)=(x)-(y);\**

**(x)=(x)-(y);**

**15、**用**两个栈实现一个队列**的功能？要求给出算法和  
思路！ 两个栈为a1 a2（默认空栈）,

一、默认两个栈都是空的

1.先将数据全部存入一个a1栈中，

2. 将a1的数据全部放进a2栈，然后出栈

二、默认两个栈中都有数据，

1.先将a2的数据存入到a1中，这时a2是空的

2.将a1的数据出栈到a2，最后一个元素作为出队的元素

3,按1—2步操作，将栈中的数据清空

4.再将数据全部存入一个a1栈中，

5.将a1的数据全部放进a2栈，然后出栈

两个空队列（q1，q2） 实现一个栈:

1. 先将数据全部放入q1队列中，然后将数据 除最后一个元素 放入到最后一个q2队列中，最后一个数据作为出栈元素
2. 然后再将q2的队列 除最后一个元素 放入q1队列中，再将最后一个元素作为出栈元素

3.执行上述操作，元素清空

16、请简述以下两个for  循环的优缺点。

//循环1:

for (i=0; i<N; i++)

{

if (condition)

DoSomething();

else

DoOtherthing();

}

//循环2：

if (condition)

{

for (i=0; i<N; i++)

DoSomething();

}

else

{

for (i=0; i<N; i++)

DoOtherthing();

}

第一个：优点:程序简洁，即使dosomething()或dootherthing()这2个函数中改变了condition的值，每次循环都会判断conditon , 依旧会按照我的意图执行

缺点:每一次循环，都要判断一次，时间复杂度比较差（运行时效率）

第二个：优点，只会判断一次，执行对应的循环，时间复杂度比价好，适合condition不改变的情况

缺点:只会执行一个函数，不是DoSomething();就是DoOtherthing()，如果在这两个函数中改变了condition的值，程序依旧不会改变所执行的函数，牺牲了弹性

17、unsigned char \*p1; 1byte  
unsigned long \*p2; 4byte  
p1=(unsigned char \*)0x801000;  
p2=(unsigned long \*)0x810000;  
请问p1+5=0x801005 ;p2+5= 0x810014;

18、如何避免头文件被重复包含？

#ifndef 宏名

#define 宏名

#endif

19、const 符号常量；说明上面三种描述的区别；  
(1)const char \*p –》常指针变量 ： 指向的空间不能改变  
(2)char const \*p –》 常指针变量 ： 指针指向空间不能变  
(3)char \* const p –》 指针常变量 ： 变量的空间不能发生改变

20、进程间通信的方式有?

1.管道通信🡪 匿名管道，命名管道

2. 信号

3．共享内存

4，消息队列

5.socket

21、sizeof与strlen的区别

Sizeof 是运算符  
 strlent 是函数

Sizeof 求空间长度大小（以类型、函数、做参数）

Strlen 求字符串长度,遇到’\0’停止（只能以char\*做参数）

22、试写出a变量与“零值”的比较。

1、若a是int类型

0==a; 0!=a

2、若a是bool类型

False == a;(错误，因为每一个软件中True 可能的值可能不一样，vc++ 1,vb 是-1) !a a

3、若a是float类型 Ps:精度问题

0.0==a; 0.0！=a(错误)

-0.00001 <= x <=0.00001

const float EPSINON = 0.00001;

if ((x >= -EPSINON) && (x <= EPSINON)

4、若a是指针类型

NULL == a; NULL!=a;

23、①判断一段程序是由C 编译程序还是由C++编译程序编译

通过编译的命令是为gcc 还是 g++ (gcc为c的编译命令，g++为c++的编译命令)

判断\_cplusplus (c++)或\_STDC\_ (c)宏: 通常许多编译器还有其他编译标志宏

#ifdef \_\_cplusplus

       cout<<"c++";

#else

       cout<<"c";

#endif

  ②判断一个操作系统是16位还是32位的

16位默认 是 2byte int 2byte

32 为 默认是4byte int 4byte

定义一个指针变量，通过sizeof来判断 –》

该指针变量大小为 2byte –》16位

该指针变量大小为 4byte🡪32位

③两个类型为int的变量a,b，不使用if、?:、switch等判断语句，找出最大的变量。

利用绝对值方法

(A+b)+abs(a-b)/2 , 结果为最大值

利用位移操作符 a,b为int:

a-b>>sizeof(int)\*8-1 为 正负 1 的话说明b大，为0的话说明a大

24、在不用第三方参数的情况下，交换两个参数的值

两种方法:

1. 数学交换:

A+=b;

B=a-b;

A=a-b;

1. 异或法:

A^=b;

B^=a^b;

A=a^b;

25、C和C++有什么不同?

C语言：面向过程，C++:面向对象的

C++可以完全兼容C语言,c语言的所有东西，c++都可以用,

C++ 还额外增加了封装 继承 多态的概念

C++可以用泛型，c中没有泛型概念等等

26、“引用”与指针的区别是什么？

引用:两个变量指向同一块空间，并没有分配空间（直接访问）

引用在开始的时候就绑定到了一个内存空间(开始必须赋初值),所以他只能是这个内存空间的名字,而不能改成其他的,当然可以改变这个内存空间的值.

指针：它是指向变量空间的，他有自身的空间 (间接访问)，可以改变指向的空间

27、有了 malloc/free 为什么还要 new/delete  ？

Malloc/free：只能分配空间，无法调用对象的构造器和析构器

New/delete:申请与释放,分别调用对象的构造器和析构器

在c++中引入了类的概念，而类只能通过构造器来分配空间，

Malloc并不能调用构造器，所以无法给给对象分配空间  
，这是new的作用就展现出来了

28、内存的分配方式有几种？

栈区分配  
静态区分配  
堆区分配

29、memset、memcpy和strcpy的根本区别

Memset:给内存空间初始化的（任意类型）

Strcpy: 只能拷贝字符串

Memcpy:可以拷贝任意类型元素

30、虚函数有什么作用？

虚函数：主要用于多态，父类指针指向子类对象时，

如果是普通函数：父类指针只能引用子类继承过来的，属于自己的函数

而虚函数就可以完美解决该问题,将基类的函数定义成virtual函数,子类重写父类方法 ，这是父类指针对象引用的就是子类的函数

31、以下为Windows NT下32位C++程序，请计算sizeof的值

(1) void Func(char str[100])

{

  sizeof(str)=? 4

}

(2) void \*p=malloc(100);

sizeof(p)=? 4

(3) int a[100];

sizeof(a)=? 400

(4)char \*p=”aaaaaaa”;

sizeof(p)=? 4byte

32、C++重写、重载、重定义的区别？

重载：静态多态，同名，形参不同，返回值可以不同

重写和重定义：建立在子父类，名字和返回值、形参相同

重定义:

父类中的函数不是虚函数，子类 –》重新定义 父类中的方法，称为重定义（隐藏） ，子类中有两个该方法，一个是自己的，一个是父类的，只是父类的被隐藏了

重写:

父类中的函数时虚函数，子类重定义父类中的方法，称为重写，子类重写父类的方法，子类中只有一个属于自己的方法，父类的方法已经没有了

33、自动调用复制构造函数的几种情形 类名 box

1.在程序中需要新建一个对象，并用另一个对象，给他进行初始化  
 box b;

Box c(b);

2/当函数为类对象是

Void getObj(box b){}

Box c;

getObj(c);

1. 当返回值为对象时

Box getObj(){

Box c;

Return c;

}

33、如何重载前++和后++运算符？

Operator++(){}//前++

Operator++(int){}//后++

34、函数模板与函数重载的异同？

异:

1.函数模板:泛型，可以接收多种类型

Template<typename T>

2.函数重载:同名，不同参，称为函数重载

3. 函数模板不允许自动类型转化,函数重载能够进行自动类型转换

同: 函数模板与重载都采用静态绑定实现了编译时的多态性

35、编写一个单例模式的例子

//单实例，只能实例化一次

class game{

static game\* init()

{

if(!instance)

Instance = new game();

return instance;

}

Protected:

game(){

cout<<”初始化函数”<<endl;

}

static game\* instance;

}

game\* game::instance=NULL;//初始化静态成员函数

36、什么是多态？实现多态的两种方式。

多态：一种事物，多种形态，必须建立在继承的基础上

多态的两种方式:

基类指针指向子类

基类\* 变量名 = 子类对象地址

//一个作为参数:

一、Void getPet(基类\* 变量名)

//一个作为返回值

二、基类\* getPet()

37、定义宏MAX和MIN

#define MAX(a,b) ((a)>(b)?(a):(b))

#define MIN (a,b) ((a)<(b)?(a):(b))

四、编程题：

1. 有两个磁盘文件A和B, 各存放一行字母，要求把这两个文件中的信息合并（按字母顺序排列），输出到一个新文件C 中。

int mystrlen(const char\* str)

{

int i=0;

for(i=0;str[i]!='\0';i++);

return i;

}

char\* mystrcat(char\* des,const char\* src)

{

int i=0;

int length = mystrlen(des);

for(i=0;(des[length+i]=src[i])!='\0';i++);

return des;

}

int main()

{

char buf[100],buf2[100];

FILE\* fr = fopen("./a.txt","r");

FILE\* fw = fopen("./b.txt","r");

if(!fr||!fw)

{

perror("open fail");

return -1;

}

char\* a=fgets(buf,sizeof(buf),fr);

char\* b = fgets(buf2,sizeof(buf2),fw);

mystrcat(a,b);

FILE\* fc = fopen("./c.txt","w+");

if(!fc)

{

perror("open fail");

return -1;

}

int i=0,j=0;

int listen = 0;

for(i=0;a[i]!='\0';i++)

{

listen = i;

for(j=i+1;a[j]!='\0';j++)

{

if(a[j]<a[listen])

listen=**j**;

}

if(listen!=i)

{

int temp = a[i];

a[i]=a[listen];

a[listen]=temp;

}

}

//将排序好的a 放入到c文件中

fputs(a,fc);

fclose(fr);

fclose(fw);

fclose(fc);

return 0;

}

1. 计算字符串中子串出现的次数。

Void myfindstr(const char\* str,const char\* s)

{

Int i=0,j=0;

Int count=0;

For(i=0;str[i]!=’\0’;i++)

{

For(j=0;s[j]!=’\0’&&str[i+j]==s[j];j++);

If(s[j]==’\0’)

Count++;

}

cout<<”子串出现的次数:”<<count<<endl;

}

1. 实现strcmp 函数。

Int mystrcmp(const char\* str1,const char\* str2)

{

Int i=0;

For(i=0;\*(str1+i)!=’\0’&&\*(str1+i)==\*(str2+i);i++);

Retrun (str1[i]!=str2[i])?(str1[i]>str2[i]?1:-1):0;

}

1. 取一个整数a从右端开始的4～7位。

Int a=100;

If(a<10000000)

{

Cout<<”输入错误”<<endl;

}

a /= 1000;

a%=10000;

cout<<a<<endl;

写一个内存拷贝函数memcpy(void\*src,void\*des),不用任何库函数.

void\* mymemcpy(void\* des ,const void\* src , int n)

{

if(des ==NULL || src == NULL)

{

return NULL;

}

char\* pdes = (char\*)des;

char\* psrc = (char\*)src;

int i=0;

while(i<n)

{

\*(pdes+i)=\*(psrc+i);

i++;

}

return pdes;

}

1. 把一个单向链表链表反转。void reverse(test\* head)

void reverse(test\* head)

{

Test\* ploc = head;

Test\* temp = NULL;

Head =NULL;

While(ploc)

{

Temp = ploc;

Ploc = ploc->next;

Temp->next = head;

Head = temp;

}

}

1. 华为面试题：怎么判断链表中是否有环？

if(head==NULL)//无环

return false;

//定义两个指针，都记录头节点

ListNode\* pHeadOne = head;//慢指针--》一次走一格

ListNode\* pHeadTwo = head;//快指针--》一次走两格

//pHeadTwo 一定先比 pHead 先为NULL

while(pHeadTwo!=NULL&&pHeadTwo->next!=NULL&&pHeadTwo->next->next!=NULL)

{

pHeadOne = pHeadOne->next;

pHeadTwo = pHeadTwo->next->next;

if(pHeadOne == pHeadTwo )

return true;

}

return false;

1. 请编写一个C 函数，该函数将给定的一个整数转换成字符串。

int mystrlen(const char\* str)

{

int i=0;

for(i=0;str[i]!='\0';i++);

return i;

}

char\* myitoa(int a)

{

char buf[100];

char buf2[100];

char temp='\0';

int i=0;

if(a<0)

{

temp='-';

a=a\*(-1);

}

for(;a!=0;a/=10)

{

buf[i++]=(char)((a%10)+'0');

}

buf[i]=temp;

i=0;

int length = mystrlen(buf);

for(length-=1;length>=0;length--)

{

buf2[i++]=buf[length];

}

buf2[i]='\0';

return buf2;

}