C++笔记

Q1.void*大小?

void* 可以看做一个integeral type.
sizeof(void*)是取决于这个整形量有多少个字节而不是取决于其指向了啥决定的。
void*有多大看编译器目标平台类型。x86通常4 bytes. x64通常8bytes

Q2.

请问下面的程序一共输出多少个“-”？为什么？

①
```
int main(void) {
   int i;
   for (i=0; i<2; i++) {
      fork();
      printf("-\n");
   }
   return 0;
}
```

②

```
int main(void) {
   int i;
   for (i=0; i<2; i++) {
      fork();
      printf("-");
   }
   return 0;
}
```

答案：①为6个，②为8个。
首先fork()是可以复制父进程的缓存，变量值等信息.
①
i = 0时，父进程A产生一个子进程A1，此时输出两行“ – ”;
i = 1时，fork使父进程A产生子进程A2，A1产生子进程A3，此时A – A3共产生4行“ – ”（因为现在A，A1的输出行缓冲均为空）；

总数为6：2（A）+2（A1）+1（A2）+1（A3）=6；
若还能循环一次，则3（A）+3(A1)+2(A2)+2 (A3)+4=14.

②I=0时，父进程A产生一个子进程A1，此时不输出！！因为没有\n清除缓冲区，相当于子进程也复制了父进程的缓冲区，即里面有一个’-’
假设父A0,子A1，现在缓冲区各有一个’-‘。
I =1时，A1产生A3，A2产生A4，各有两个’A’.共有8个
若还能循环一次,应该为8 * 3 = 24个。

Q3

动态联编 只能通过指针或引用标识对象来操作虚函数 ;如果采用一般的标识对象来操作虚函数,将采用静态联编的方式调用虚函数

Q4.

以下代码输出什么__.

```
main()
{
   int a[5]={1,2,3,4,5};
   int *p=(int *)(&a+1);
   printf("%d",*(p-1));
}
```

答案：5
第二句话将p赋值为一个指针。&a+1意思是a这个指针地址的下一位，即指向下一个数组的指针（这个数组未定义），但是p-1指向的就是a中的最后一位，即5.

Q5

非虚函数子类是不回调用的，如：

```
class A
{
    public:
    ~A();
};
A::~A()
```

```
{
    printf("delete A ");
}
class B : public A
{
    public:
    ~B();
};
B::~B()
{
    printf("delete B ");
}
请问执行以下代码
A *pa = new B();
delete pa;
```

输出的只有

```
printf("delete A ");
```

因为A的构造函数不是虚函数。

```
using namespace std;
class A{
    public:
        virtual void f() { cout << "A::f() "; }
        void f() const { cout << "A::f() const "; }
};
class B : public A {
    public:
        void f() { cout << "B::f() "; }
        void f() const { cout << "B::f() const "; }
};
void g(const A* a) {
    a->f();
}
int main(int argc, char *argv[]) {
    A* p = new B();
    p->f();
    g(p);
    delete(p);
    return 0;
}
```

输出应该为

```
B::f() A::f() const
```

因为const函数未被定义为虚函数，所以调用基类函数。

Q6

```
template<class T> class Foo{
        T tVar;
    public:
        Foo(T t) : tVar(t) { }
};

template<class T> class FooDerived:public Foo<T>
{
};

int main()
{
    FooDerived<int> d(5);
    return 0;
}
```

该代码无法编译，编译错误，FooDerived是一个继承模板类的非模板类，它的类型不能改变。
当基类构造函数需要外部传递参数才能进行初始化时，派生类必须显式定义构造函数，为基类传递参数；基类如果不需要传递或者可以不传递参数，派生类可以不用显式定义构造函数。

Q7

```
char* s1 = "Hello world";
char s2[] = "Hello world";
s1[2] = 'E';     // 1
s2[2] = 'E';     // 2
*(s1 + 2) = 'E';  // 3
*(s2 + 2) = 'E';  // 4
```

语句1/3均不能执行，因为是常量。

Q8

extern ： 外部变量，可供所以源文件使用

register ： 寄存器变量，放在寄存器而非内存中，效率更高，一般是临时变量

auto :自动变量，所有未加 **static** 关键字的都默认是 **auto** 变量，也就是我们普通的变量

static ： 静态变量，在内存中只存在一个，可供当前源文件的所有函数使用

Q9

```
main() {
 int i=0;
 switch(i) {
   case 0:
     i++;
     printf("%d..",i);
   case 1:
     printf("%d..",i);
   case 2:
      printf("%d..",i);
    }
}
```

一旦case匹配成功，直到遇到break，一直会执行（包括dafault）因此会输出

```
1..1..1..
```

Q10

在linux下64位c程序，请计算输出的三个sizeof分别是（）

```
void func(char str_arg[100])
{
     cout<<sizeof(str_arg)<<endl;
}
int main(int argc,char* argv[])
{
     char str[]="Hello";
     char *p=str;
     cout<<sizeof(str)<<endl;
     cout<<sizeof(p)<<endl;
     func("test");
     return 0;
}
```

这里的sizeof(p)输出的指针的长度，64位系统中指针长度为8.因此为6，8，8.

Q11

引用只是一个别名，是已有变量的别名，而void类型是空类型，是没有分配内存的。所以引用不能是void类型。

Q12

下面的程序输出可能是什么？

```
class Printer{
    public:
        Printer(std::string name) {std::cout << name;}
};
class Container{
    public:
        Container() : b("b"), a("a") {}
    Printer a;
    Printer b;
};
int main(){
    Container c;
    return 0;
}
```

派生类实例化时，先调用基类的构造函数，然后是派生类的类成员变量构造函数（构造的顺序是按照成员变量的定义先后顺序，而不是按照初始化列表的顺序），最后是派生类的构造函数。
由于a现被定义，因此会输出ab。

Q13

```
#define MAX 255
int main()
{
     unsigned char A[MAX], i;
     for (i = 0; i <= MAX; i++)
         A[i] = i;
}
```

这里会数组越界且死循环，因为unsigned char 最大为255..当255时i++变成了0.

Q14

```
int i = 5;
int j = 10;
System.out.println(i + ~j);
```

答案为-6
公式-n=~n+1可推出~n=-n-1，所以~10=-11再加5结果为-6

Q15

进程与应用程序的区别在于应用程序作为一个静态文件存储在计算机系统的硬盘等存储空间中，而进程则是处于动态条件下由操作系统维护的系统资源管理实体。

Q16

在64位系统下，分别定义如下两个变量：char *p[10]; char(*p1)[10];请问，sizeof(p)和sizeof (p1)分别值为__。
80，8.
char *p[10]为大小为10的指针数组，64位每个指针大小为8，因此总共为80.
char (*p1)[10]是数组指针。大小为8.

Q17

对一批编号为1～100，全部开关朝上(开)的灯进行以下操作：凡是1的倍数反方向拨一次开关；2的倍数反方向又拨一次开关；3的倍数反方向又拨一次开关……100的倍数反方向又拨一次开关 ,问：最后为关熄状态的灯有几个。

若一个开关编号为x，那么所有能整除x的数，都会导致开关拨动一次，例如 x = 8时，当每次拨动1,2,4,8的时候，都会导致8号开关拨动一次。
那么只有当能整除x的数为奇数的时候，最后开关才会处于关闭(因为一开始是开着的，经过奇数次拨动就处于关闭）
那么什么样的数才能满足？答案是若x是某个数的平方数的时候。因为 一般情况下，如果 x/a = b, 那么 a，b肯定会作为x的2个因数，所以一般情况下，肯定是成对出现的，只有当a=b的时候，这样才会出现奇数个因数。

Q18

产生死锁的四个必要条件：
（1） 互斥条件：一个资源每次只能被一个进程使用。
（2） 请求与保持条件(占有等待)：一个进程因请求资源而阻塞时，对已获得的资源保持不放。
（3） 不剥夺条件:进程已获得的资源，在末使用完之前，不能强行剥夺。
（4） 环路等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。

Q19

java中
类只能继承一个父类，但是可以实现多个接口，**对**

抽象类自身可以定义成员而接口不可以，**错，接口允许定义成员，但必须是常量**。

抽象类和接口都不能被实例化 **对**

一个类可以有多个基类和多个基接口，**错，java中一个基类**

Q20

HTTPS在传输数据之前需要客户端（浏览器）与服务端（网站）之间进行一次握手，在握手过程中将确立双方加密传输数据的密码信息。TLS/SSL协议不仅仅是一套加密传输的协议，更是一件经过艺术家精心设计的艺术品，TLS/SSL中使用了非对称加密，对称加密以及HASH算法。握手过程的简单描述如下：
1.浏览器将自己支持的一套加密规则发送给网站。
2.网站从中选出一组加密算法与HASH算法，并将自己的身份信息以证书的形式发回给浏览器。证书里面包含了网站地址，加密公钥，以及证书的颁发机构等信息。
3.获得网站证书之后浏览器要做以下工作：
a) 验证证书的合法性（颁发证书的机构是否合法，证书中包含的网站地址是否与正在访问的地址一致等），如果证书受信任，则浏览器栏里面会显示一个小锁头，否则会给出证书不受信的提示。
b) 如果证书受信任，或者是用户接受了不受信的证书，浏览器会生成一串随机数的密码，并用证书中提供的公钥加密。
c) 使用约定好的HASH计算握手消息，并使用生成的随机数对消息进行加密，最后将之前生成的所有信息发送给网站。
4.网站接收浏览器发来的数据之后要做以下的操作：
a) 使用自己的私钥将信息解密取出密码，使用密码解密浏览器发来的握手消息，并验证HASH是否与浏览器发来的一致。
b) 使用密码加密一段握手消息，发送给浏览器。
5.浏览器解密并计算握手消息的HASH，如果与服务端发来的HASH一致，此时握手过程结束，之后所有的通信数据将由之前浏览器生成的随机密码并利用对称加密算法进行加密。
这里浏览器与网站互相发送加密的握手消息并验证，目的是为了保证双方都获得了一致的密码，并且可

以正常的加密解密数据，为后续真正数据的传输做一次测试。另外，HTTPS一般使用的加密与HASH算法如下：
非对称加密算法：RSA，DSA/DSS
对称加密算法：AES，RC4，3DES
HASH算法：MD5，SHA1，SHA256

Q21

下列关于线程调度的叙述中,错误的是()
正确答案: B C

A.调用线程的sleep()方法,可以使比当前线程优先级低的线程获得运行机会

B.调用线程的yeild()方法,只会使与当前线程相同优先级的线程获得运行机会（**yield()方法使得当前的线程让出CPU的使用权，以使得比该线程优先级相同或更高的线程有机会运行。该线程在让出CPU使用权之后可能再次被选中，因此yield()方法可能会不起作用(这也说明了yield()方法不会使得比当前线程优先级低的线程运行)。**）

C.具有相同优先级的多个线程的调度一定是分时的（**java虚拟机中如果多个线程优先级相同，则会随机选择一个线程占用CPU，处于运行状态的线程会一直运行，直至它不得不放弃CPU为止，因此不一定是分时调度。**）

D.分时调度模型是让所有线程轮流获得CPU使用权

Q22

include命令的功能是（）:在命令处插入一个文本文件.

Q23

```
#define M(x,y,z) x*y+z
main()
{
	int a=1, b=2, c=3;
	printf("%d/n",M(a+b,b+c,c+a));
}
```

因为这里没有加括号

```
#define M(x,y,z) x * y + z
//代入
a + b * b + c + c + a
```

正确的应该是

```
#define M(x,y,z) (x) * (y) + z
```

所以不要忘记括号!

Q24

```
#include<iostream>
using namespace std;
class B0//基类BO声明
{
public://外部接口
virtual void display()//虚成员函数
{
    cout<<"B0::display0"<<endl;}
};
class B1:public B0//公有派生
{
public:
    void display() { cout<<"B1::display0"<<endl; }
};
class D1: public B1//公有派生
{
public:
    void display(){ cout<<"D1::display0"<<endl; }
};
void fun(B0 ptr)//普通函数
{
    ptr.display();
}
int main()//主函数
{
    B0 b0;//声明基类对象和指针
    B1 b1;//声明派生类对象
    D1 d1;//声明派生类对象
    fun(b0);//调用基类B0函数成员
    fun(b1);//调用派生类B1函数成员
    fun(d1);//调用派生类D1函数成员
}
```

由于这里的fun函数传递的是值，所以不触发多态，只有指针和引用才会触发多态。

Q25

```
int main(int argc, char *argv[])
{
    string a="hello world";
    string b=a;
    if (a.c_str()==b.c_str())
    {
        cout<<"true"<<endl;
    }
    else cout<<"false"<<endl;
    string c=b;
    c="";
    if (a.c_str()==b.c_str())
    {
        cout<<"true"<<endl;
    }
    else cout<<"false"<<endl;
    a="";
    if (a.c_str()==b.c_str())
    {
        cout<<"true"<<endl;
    }
    else cout<<"false"<<endl;
    return 0;
}
```

c_str()函数返回的是一个指针，比较指针当然不同了，因此全部是FALSE；

Q26

```
 有一个类B继承自类A，他们数据成员如下：
class A {
...
private:
     int a;
};
class B   : public A {
...
private:
```

```
    int a;
public:
    const int b;
    A &c;
    static const char* d;
    B* e;
}
```
则构造函数中，成员变量一定要通过初始化列表来初始化的是____。

构造函数初始化时必须采用初始化列表一共有三种情况，
1.需要初始化的数据成员是对象(继承时调用基类构造函数)
2.需要初始化const修饰的类成员
3.需要初始化引用成员数据

Q27

多线程调用时要进行保护时，主要是针对全局变量和静态变量的，

Q28

```
#include <stdio.h>
#include <stdio.h>
void fun( char *s )
{
    char a[10];
    strcpy ( a, "STRING" );
    s = a ;
}
main( )
{
    char *p= "PROGRAM" ;
    fun( p );
    printf ( "%s\n ", p) ;
}
```

最终会输出PROGRAM.本题考查字符串指针作为函数参数,本题中p作为字符串指针传入fun中,p指向的内容并没有发生变化,所以选项D正确。也就是说指向内存块的指针变了，本身的字符串没有变。

Q29

c++中，声明const int i,是在**编译**阶段做到 i只可读的。

Q30

```
typedef struct
{
char flag[3];
short value;
} sampleStruct;
union
{
char flag[3];
short value;
} sampleUnion;
```

假设 **sizeof**(**char**)=1,**sizeof**(**short**)=2,那么**sizeof**(sampleStruct) = 1 ,
sizeof(sampleUnion) = 2

本题的考点在于结构体的对其。
字符型占用1字节，不需要字节对齐
short占用2字节，需要两字节对齐
所以sizeof(sampleStruct) = 3 *（1字节）+（1个补齐字节）+1*（2字节） = 6
联合体
占用大小采用成员最大长度的对齐，最大长度是short的2字节
但 char flag[3]需要3个字节
所以 sizeof(sampleUnion) = 2*（2字节） = 4

对Union结构体，sizeof的取值不仅考虑sizeof最大的成员，还要考虑对齐字节，对齐字节的取值是取成员类型字节最大值与指定对齐字节(32位机器默认是4，64位机器默认是8)两者中的较小值，本题中成员类型最大值为short，2，与指定对齐字节4比较取2，所以要2字节对齐，如果去掉short变量，则取char的1字节对齐，结果为3.