

2019数组、结构体练习

2-1设有数组定义 `char array []="China"`; 则数组 `array`所占的空间为 (C)

- A. 4个字节
- B. 5个字节
- C. 6个字节
- D. 7个字节

2-2数组定义为 `int s[3][2]={1, 2, 3, 4, 5, 6}`, 数组元素 (B) 的值为4

- A. `s[3][0]`
- B. `s[1][1]`
- C. `s[1][2]`
- D. `s[0][1]`

2-3对于以下结构定义, `++p->str` 中的 `++` 加在(C)

```
struct {  
    int len;  
    char *str;  
} *p;
```

A. 指针 `str` 上 B. 指针 `p` 上 C. `str` 指的内容上 D. 以上均不是

2-4以下哪个选项中的 `p` 不是指针(C)

- A. `int **p`
- B. `int (*p)[5];`
- C. `int *p[6];`
- D. `struct stu{`

`char name[20];`

`int age; }*p, q;`

2-6定义如下结构体:

```
struct sk  
{  
    int a;  
    float b;  
}data, *p;
```

若指针 `p` 指向结构体变量 `data`, 即有 `p = &data;`, 则对结构体成员 `a` 的正确引用(B)

- A. `(*p).data.a`
- B. `(*p).a`

C. `p->data.a`

D. `p.data.a`

2-7下面的这个循环的循环次数是 (B)

`for(int i=0,j=10; i=j=10; i++,j--)`

A.语法错误, 不能执行

B. 无限次

C. 10

D. 1

2-9有以下函数: `char fun(char *p) { return p; }` 该函数的返回值是(A)

A.无确切的值

B.形参 p 中存放的地址值

C.一个临时存储单元的地址

D.形参 p 自身的地址值

2-10以下不正确的赋值或赋初值的方式是(C)

A. `char str[]="string";`

B. `char str[7]={'s', 't', 'r', 'i', 'n', 'g'};`

C. `char str[10]; str="string";`

D. `char str[7]={'s', 't', 'r', 'i', 'n', 'g', '\0'};`

2-11以下代码:

```
struct Student{
    int n;
    struct Student * next;
};
struct Student a[3]={5,&a[1],7,&a[2],9, NULL};
struct Student *p;
p=a;
```

那么, 以下表达式不能够正确地访问到第3个结构体数组元素a[2]的成员n (其值为9) 的是(C)

A. `p[2].n`

B. `(p+2)->n`

C. `*(p+2).n`

D. `p->next->next->n`

2019指针练习

2-1若 p1、p2 都是整型指针, p1 已经指向变量 x, 要使 p2 也指向 x, (A)是正确的

A. `p2 = p1;`

B. `p2 = **p1;`

C. `p2 = &p1;`

D. `p2 = *p1;`

2-2若已定义: `int a[9]`, `*p=a;` 并在以后的语句中未改变 `p` 的值, 不能表示 `a[1]` 地址的达式是(C)

A. `p+1`

B. `a+1`

C. `a++`

D. `++p`

2-3下列程序的输出是 (D)

```
#include<stdio.h>
int main(void)
{
    int a[12] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 }, *p[4], i;

    for (i = 0; i < 4; i++)
        p[i] = &a[i*3];
    printf("%d\n", p[3][2]);

    return 0;
}
```

A.上述程序有错误

B.6

C.8

D.12

2-4存在定义 `int a[10]`, `x`, `*pa;` , 若 `pa=&a[0]` , 下列的哪个选项和其他3个选项不是等价的(A)

A. `x=*pa;`

B. `x=*(a+1);`

C. `x=*(pa+1);`

D. `x=a[1];`

2-5 `int *p` 的含义是(B)

A. `p` 是一个指针, 用来存放一个整型数

B. `p` 是一个指针, 用来存放一个整型数据在内存中的地址

C. `p` 是一个整型变量

D. 以上都不对

2-6以下哪个选项中的 `p` 不是指针(C)

A. `int **p`

B. `int (*p)[5];`

C. `int *p[6];`

```
D. struct Stu{  
    char name[20];  
    int age; }*p, q;
```

2-7设有下面的程序段: `char s[]="china"; char *p=s;` 则下列叙述正确的是(D)

- A. 数组s中的内容和指针变量p中的内容相等
- B. s和p完全相同
- C. s数组长度和p所指向的字符串长度相等
- D. *p与s[0]相等

2-8在基类型相同的两个指针变量之间, 不能进行的运算是(C)

- A. <
- B. =
- C. +
- D. -

2-9已知 `int a, *pa=&a`, 输出指针pa 十进制的地址值的方法是()

- A. `cout<<pa;`
- B. `cout<<&pa;`
- C. `cout<<*pa;`
- D. `cout<<long(&pa);`

2-11在说明语句 `int *f()`; 中, 标识符 f 代表是(D)

- A. 一个用于指向整型数据的指针变量
- B. 一个用于指向一维数组的行指针
- C. 一个用于指向函数的指针变量
- D. 一个返回值为指针型的函数名

2-12若有定义: `int (*p)[4];` 则标识符 p (C)

- A. 是一个指向整型变量的指针
- B. 是一个指针数组名
- C. 是一个指针, 它指向一个含有四个整型元素的一维数组
- D. 定义不合法

2-13根据声明 `int a[10], *p=a;`, 下列表达式错误的是(C)

- A. `a[9]`
- B. `p[5]`
- C. `a++`
- D. `*p++`

2-14 `const int *p` 说明不能修改(D)

- A.p指针;
- B.p指针指向的变量;
- C.p指针指向的数据类型;
- D.上述A、B、C三者;

2-15以下哪个定义中的 p 不是指针, 请选择恰当的选项()

- A. `char **p;`
- B. `char (*p)[10];`
- C. `char *p[6];`
- D.给出的三项中, p 都是指针

2-17根据声明 `int (*p)[10]`, p 是一个()

- A.数组
- B.指针
- C.函数
- D.数组的元素

2-18若变量已正确定义并且指针p已经指向某个变量x, 则`(*p)++`相当于(B)

- A. `p++`
- B. `x++`
- C. `*(p++)`
- D. `&x++`

2019链表练习

2-2在一个以 h 为头的单向链表中, p 指针指向链尾的条件是(B)

- A. `p->next=h`
- B. `p->next=NULL`
- C. `p->next->next=h`
- D. `p->.data=-1`

2-3对于一个头指针为head的带头结点的单链表, 判定该表为空表的条件是(B)

- A. `head==NULL`
- B. `head->next==NULL`
- C. `head->next==head`
- D. `head!=NULL`

2-4在一个单链表head中, 若要在指针p所指结点后插入一个q指针所指结点, 则执行()

- A. `p->next=q->next; q->next=p;`
- B. `q->next=p->next; p=q;`

C. `p->next=q->next; p->next=q;`

D. `q->next=p->next; p->next=q;`

2-5以下程序的输出结果是(A)

```
struct HAR
{int x, y; struct HAR *p;} h[2];
int main(void)
{
    h[0].x=1; h[0].y=2;
    h[1].x=3; h[1].y=4;
    h[0].p=h[1].p=h;
    printf("%d%d\n", (h[0].p)->x, (h[1].p)->y);
    return 0;
}
```

A.12

B.23

C.14

D.32

2-6以下程序的输出结果是(D)

```
struct NODE{ int num; struct NODE *next; };
int main(void)
{ struct NODE *p,*q,*r;
  p=(struct NODE*)malloc(sizeof(struct NODE));
  q=(struct NODE*)malloc(sizeof(struct NODE));
  r=(struct NODE*)malloc(sizeof(struct NODE));
  p->num=10; q->num=20; r->num=30;
  p->next=q;q->next=r;
  printf("%d\n",p->num+q->next->num);
  return 0; }
```

A.10

B.20

C.30

D.40

2-7下面程序段输入一行字符，按输入的逆序建立一个链表(B)

```

struct node{
    char info;
    struct node *link;
} *top, *p;
char c;
top=NULL;
while((c=getchar())!='\n')
{
    p=(struct node*)malloc(sizeof(struct node));
    p->info=c;
    top=p;
}

```

A. `top->link=p`

B. `p->link=top`

C. `top=p->link`

D. `p=top->link`

2-8在单链表指针为p的结点之后插入指针为s的结点，正确的操作是(B)

A. `p->next=s;s->next=p->next;`

B. `s->next=p->next;p->next=s;`

C. `p->next=s;p->next=s->next;`

D. `p->next=s->next;p->next=s;`

2-10在一个单链表head中，若要删除指针p所指结点的后继结点，则执行(D)

A. `p=p->next;free(p);`

B. `p->next=p->next->next; free(p);`

C. `q= p->next q->next=p->next; free(q);`

D. `q=p->next; p->next=q->next; free(q);`

2-11以下结构类型可用来构造链表的是(B)

A. `struct aa{ int a; int * b; };`

B. `struct bb{ int a; bb * b; };`

C. `struct cc{ int * a; cc b; };`

D. `struct dd{ int * a; aa b; };`

2-12设单链表中结点的结构为

```

struct LinkNode{ //链表结点定义
    E data; //数据
    LinkNode * link; //结点后继指针
};

```

不带头结点的单链表first为空的判定条件是(A)

A. `first==NULL;`

B. `first->link== NULL;`

C. `first->link== first;`

D. `first! = NULL;`

2-13可以用带头附加结点的链表表示线性表，也可以用不带头结点的链表表示线性表，前者最主要的好处是**(B)**

A.可以加快对表的遍历

B.使空表和非空表的处理统一

C.节省存储空间

D.可以提高存取表元素的速度

2-14已知L是带头结点的单链表，则摘除首元结点的语句是**(B)**

A. `L=L->link;`

B. `L->link=L->link->link;`

C. `L=L->link->link;`

D. `L->link = L;`

2-15单链表中，增加一个头结点的目的是**(C)**

A.使单链表至少有一个结点。

B.标识表结点中首结点的位置。

C.方便运算的实现。

D.说明单链表是线性表的链式存储。

2-16需要分配较大空间，插入和删除不需要移动元素的线性表，其存储结构是**(B)**

A.单链表

B.静态链表

C.线性链表

D.顺序存储结构

2019类和对象练习

2-1关于delete运算符的下列描述中，**(C)**是错误的

A.它必须用于new返回的指针；

B.使用它删除对象时要调用析构函数；

C.对一个指针可以使用多次该运算符；

D.指针名前只有一对方括号符号，不管所删除数组的维数。

2-3域作用符“::”用于：**(D)**

A.限定在类外定义的成员函数

B.访问定义于某个命名空间的对象

C.访问类的静态成员

D.以上答案都正确

2-4建立一个类对象时，系统自动调用(A)

- A.构造函数
- B.析构函数
- C.友元函数
- D.成员函数

2-6以下说法正确的是(C)

- A.每个对象内部都有成员函数的实现代码
- B.一个类的私有成员函数内部不能访问本类的私有成员函数
- C.类的成员函数之间可以相互调用
- D.编写一个类时，至少要写一个成员函数

2-7如果某函数的返回值是一个对象，则该函数被调用时，返回的对象(A)

- A.是通过复制构造函数初始化的
- B.是通过无参数的构造函数初始化的
- C.用哪个构造函数初始化取决于函数中return语句是怎么写的
- D.不需要初始化

2-8以下关于this指针的说法不正确的是(B)

- A. const 成员函数内部不可以使用this指针
- B.成员函数内的this指针指向成员函数所作用的对象
- C.在构造函数内部可以使用this指针
- D.在析构函数内部可以使用this指针

2-9在下面类声明中，关于生成对象不正确的是(C)

```
class point
{ public:
    int x;
    int y;
    point(int a,int b)
    {x=a; y=b;}
};
```

- A. point p(10,2);
- B. point *p=new point(1,2);
- C. point *p=new point[2];
- D. point *p[2]={new point(1,2), new point(3,4)};

2-10下面关于静态成员的描述中，正确的是 (A)

- A.静态数据成员是类的所有对象共享的数据
- B.类的每个对象都有自己的静态数据成员
- C.类的不同对象有不同的静态数据成员值

D.静态数据成员不能通过类的对象访问

2-11 C++语言对C语言做了很多改进，C++语言相对于C语言的最根本的变化是(D)

A.增加了一些新的运算符

B.允许函数重载，并允许设置缺省参数

C.规定函数说明符必须用原型

D.引进了类和对象的概念

2-12 下面对静态数据成员的描述中，正确的是(D)

A.静态数据成员可以在类体内进行初始化

B.静态数据成员不可以被类的对象调用

C.静态数据成员不能受private控制符的作用

D.静态数据成员可以直接用类名调用

2-13 下列关于构造函数的描述中，错误的是(D)

A.构造函数可以设置默认参数；

B.构造函数在定义类对象时自动执行

C.构造函数可以是内联函数；

D.构造函数不可以重载

2-14 假定AA为一个类，a()为该类公有的函数成员，x为该类的一个对象，则访问x对象中函数成员a()的格式为(B)

A. x.a

B. x.a()

C. x->a

D. (*x).a()

2-15 下列对重载函数的描述中,(A)是错误的

A.重载函数中不允许使用默认参数

B.重载函数中编译根据参数表进行选择

C.不要使用重载函数来描述毫无相干的函数

D.构造函数重载将会给初始化带来多种方式

2-16 某类中有一个无参且无返回值的常成员函数 Show，则正确的 Show 函数原型是(B)

A. const void Show();

B. void const Show();

C. void Show() const;

D. void Show(const);

2-17 下列各类函数中，不是类的成员函数的是(C)

A.构造函数

- B.析构函数
- C.友元函数
- D.拷贝构造函数

2-18下面关于成员函数和常成员函数的程序，其主函数中错误的语句是(C)

```
#include<iostream>
using namespace std;
class MyClass {
public:
    MyClass(int x): val(x) {}
    void Set(int x) {val = x;}
    void Print()const {cout << "val=" << val << '\t';}
private:
    int val;
};
int main() {
    const MyClass obj1(10);
    MyClass obj2(20);
    obj1.Print();    //语句 1
    obj2.Print();    //语句 2
    obj1.Set(20);    //语句 3
    obj2.Set(30);    //语句 4
    return 0;
}
```

- A.语句 1
- B.语句 2
- C.语句 3
- D.语句 4

2-19为类提供对外接口的是(A)

- A.公有成员函数
- B.私有成员函数
- C.保护成员函数
- D.友元函数

2-20对类的构造函数和析构函数描述正确的是(A)

- A.构造函数可以重载，析构函数不能重载
- B.构造函数不能重载，析构函数可以重载
- C.构造函数可以重载，析构函数也可以重载
- D.构造函数不能重载，析构函数也不能重载

2019FD-He练习2

2-1若有以下语句：`static char x[]="12345"; static char y[]={'1','2','3','4','5'};`则正确的说法是(B)

A.x数组和y数组的长度相同

B.x数组的长度大于y数组的长度

C.x数组的长度小于y数组的长度

D.x数组与y数组等价

2-2在c++语言中，定义数组后，使用数组元素时，数组下标可以是(C)

A.整形常量

B.整型表达式

C.整形常量或整形表达式

D.任何类型的表达式

2-3在 `int k=10, *p=&k;` , *p的值是(C)

A.指针变量p的地址值

B.变量k的地址值

C.10

D.无意义

2-4在声明语句 `int *fun ()` ; 时， `fun` 表示(B)

A.一个用于指向函数的指针变量

B.一个返回值为指针型的函数名

C.一个用于指向一维数组的行指针

D.一个用于指向 `int` 型数据的指针变量

2-5对于指针的运算，下列说法(C)是错误的

A.可以用一个空指针赋值给某个指针

B.一个指针可以加上一个整数

C.两个指针可以进行加法运算

D.两个指针在一定条件下，可以进行相等或不相等的运算

2-6若有语句： `char *line[6]` ; 以下叙述中正确的是(A)

A.定义 `line` 是一个数组，每个数组元素是一个基类型为 `char` 为指针变量

B.定义 `line` 是一个指针变量，该变量可以指向一个长度为 6 的字符型数组

C.定义 `line` 是一个指针数组，语句中的 * 号称为间址运算符

D.定义 `line` 是一个指向字符型函数的指针

2-7对语句 `float (*pf)(float x)` 的描述，正确的是(A)

A.一个用于指向函数的指针变量

B.一个返回值为指针型的函数名

C.一个用于指向float型数据的指针数组

D.一个用于指向float型数据的指针变量

2-8关于类和对象不正确的说法是：(C)

- A.类是一种类型，它封装了数据和操作
- B.对象是类的实例
- C.一个类的对象只有一个
- D.一个对象比属于某个类

2-9下列各类函数中，不是类的成员函数的是：(C)

- A.构造函数
- B.析构函数
- C.友元函数
- D.拷贝构造函数

2-10在C++中实现封装是借助于(B)

- A.枚举
- B.类
- C.数组
- D.函数

2-11在面向对象方法中，不属于“对象”基本特点的是(A)

- A.一致性
- B.分类性
- C.多态性
- D.标识唯一性

2-12对类的构造函数和析构函数描述正确的是(A)

- A.构造函数可以重载，析构函数不能重载
- B.构造函数不能重载，析构函数可以重载
- C.构造函数可以重载，析构函数也可以重载
- D.构造函数不能重载，析构函数也不能重载

2-13(D)是给对象取了一个别名，它引入了对象的同义词

- A.结构
- B.枚举
- C.指针
- D.引用

2-14在一个函数中，要求通过函数来实现一种不太复杂的功能，并且要求加快执行速度，选用(A)

- A.内联函数
- B.重载函数
- C.递归调用

D.嵌套调用

2-15下列给字符数组进行的初始化中，正确的是(D)

A. `int N=5, b[N][N]`

B. `int a[2]={{1},{2},{3},{3}}`

C. `int c[2][]={{1,2},{3,5}}`

D. `int d[3][2]={{1,2},{3,5}}`

2-16如果有 `int x, *p; float y, *q;` 则下面操作正确的是(C)

A. `p=x`

B. `p=q`

C. `p=&x`

D. `p=&y`

2-17若定义: `string str;` 当语句 `cin>>str;` 执行时, 从键盘输入: `Microsoft Visual Studio 6.0!` 所得的结果是 `str=` (B)

A. `Microsoft Visual Studio 6.0!`

B. `Microsoft`

C. `Microsoft Visual`

D. `Microsoft Visual Studio 6.0`

2-18在c++中, 类与类之间的继承关系具有(C)

A.自反性

B.对称性

C.传递性

D.反对称性

2-19假定 AA 为一个类, `a()` 为该类公有的函数成员, `x` 为该类的一个对象, 则访问 `x` 对象中函数成员 `a()` 的格式为(B)

A. `x.a`

B. `x.a()`

C. `x->a`

D. `(*x).a()`

2-20关于对象概念的描述中, 说法错误的是(A)

A.对象就是c语言中的结构变量

B.对象代表着正在创建的系统中的—个实体

C.对象是类的一个变量

D.对象之间的信息传递是通过消息进行的

2019友元与模板上机

2-1对于以下关于友元的说法(D)

- A.如果函数fun被声明为类A的友元函数，则该函数成为A的成员函数
- B.如果函数fun被声明为类A的友元函数，则该函数能访问A的保护成员，但不能访问私有成员
- C.如果函数fun被声明为类A的友元函数，则fun的形参类型不能是A
- D.以上答案都不对

2-2下面关于友元的描述中，错误的是(D)

- A.友元函数可以访问该类的私有数据成员
- B.一个类的友元类中的成员函数都是这个类的友元函数
- C.友元可以提高程序的运行效率
- D.类与类之间的友元关系可以继承

2-3已知类A是类B的友元，类B是类C的友元，则(D)

- A.类A一定是类C的友元
- B.类C一定是类A的友元
- C.类C的成员函数可以访问类B的对象的任何成员
- D.类A的成员函数可以访问类B的对象的任何成员

2-4不属于类的成员函数的是(C)

- A.构造函数
- B.析构函数
- C.友元函数
- D.复制构造函数

2-5若类A被说明成类B的友元，则(D)

- A.类A的成员即类B的成员
- B.类B的成员即类A的成员
- C.类A的成员函数不能访问类B的成员
- D.类B不一定是类A的友元

2-6下列的模板说明中，正确的是(C)

- A. `template < typename T1, T2 >`
- B. `template < class T1, T2 >`
- C. `template < typename T1, typename T2 >`
- D. `template (typedef T1, typedef T2)`

2-7假设有函数模板定义如下: `template Max(T a, T b ,T &c) { c = a + b ; }` 下列选项正确的是(B)

- A. `int x, y; char z ;Max(x, y, z) ;`
- B. `double x, y, z ;Max(x, y, z) ;`

C. `int x, y; float z; Max(x, y, z);`

D. `float x; double y, z; Max(x, y, z);`

2-8关于类模板，描述错误的是(A)

A. 一个普通基类不能派生类模板

B. 类模板可以从普通类派生，也可以从类模板派生

C. 根据建立对象时的实际数据类型，编译器把类模板实例化为模板类

D. 函数的类模板参数需生成模板类并通过构造函数实例化

2-9有函数模板定义如下：

```
template<typename T>
Max(T a, T b, T& c){c = a + b ;}
```

则下列调用中正确的是(B)

A. `int x, y; char z; Max(x, y, z);`

B. `double x, y, z; Max(x, y, z);`

C. `int x, y; float z; Max(x, y, z);`

D. `float x, double y, z; Max(x, y, z);`

2-10下列关于模板的说法中，错误的是(C)

A. 用模板定义一个对象时，不能省略参数

B. 类模板只能有虚拟参数类型

C. 类模板的成员函数都是模板函数

D. 类模板在编译中不会生成任何代码

2-11下面对模板的声明，正确的是(C)

A. `template<T>`

B. `template<class T1, T2>`

C. `template<class T1, class T2>`

D. `template<class T1; class T2>`

2-12下面关于继承和派生的构造函数和析构函数的程序，输出结果是(B)

```
#include<iostream>
using namespace std;
class AA {
public:
    AA() { cout << "A"; }
    ~AA() { cout << "a"; }
};
class BB: public AA {
    AA aa;
public:
    BB() { cout << "B"; }
    ~BB() { cout << "b"; }
```



```
};  
int main() {  
    BB bb;  
    return 0;  
}
```

A.AABaab

B.AABbaa

C.BAAaab

D.BAAbaa

2-13在c++中，类之间的继承关系具有(C)

A.自反性

B.对称性

C.传递性

D.反对称性

2-14下列关于类的继承描述中,(D)是正确的

A.派生类公有继承基类时，可以访问基类的所有数据成员，调用所有成员函数。

B.派生类也是基类，所以它们是等价的。

C.派生类对象不会建立基类的私有数据成员，所以不能访问基类的私有数据成员。

D.一个基类可以有多个派生类，一个派生类可以有多个基类。

2-15下列关于运算符重载的描述中，(D)是正确的

A.运算符重载可以改变操作数的个数

B.运算符重载可以改变优先级

C.运算符重载可以改变结合性

D.运算符重载不可以改变语法结构

2-16为了能出现在赋值表达式的左右两边，重载的 [] 运算符应定义为(B)

A. `A operator [] (int);`

B. `A& operator [] (int);`

C. `const A operator [] (int);`

D.以上答案都不对

2019友元与模板练习

2-1友元的作用是(A)

A.提高程序的运用效率

B.加强类的封装性

C.实现数据的隐藏性

D.增加成员函数的种类

2-2下列各类函数中，不是类的成员函数的是(C)

- A.构造函数
- B.析构函数
- C.友元函数
- D.拷贝构造函数

2-3下面关于友元的描述中，错误的是(D)

- A.友元函数可以访问该类的私有数据成员
- B.一个类的友元类中的成员函数都是这个类的友元函数
- C.友元可以提高程序的运行效率
- D.类与类之间的友元关系可以继承

2-4下面对于友元函数描述正确的是(C)

- A.友元函数的实现必须在类的内部定义
- B.友元函数是类的成员函数
- C.友元函数破坏了类的封装性和隐藏性
- D.友元函数不能访问类的私有成员

2-5对于类之间的友元关系(D)

- A.如果类A是类B的友元，则B的成员函数可以访问A的私有成员
- B.如果类A是类B的友元，则B也是A的友元。
- C.如果类A是类B的友元，并且类B是类C的友元，则类A也是类C的友元。
- D.以上答案都不对。

2-6现有声明：

```
template  
class Test{...};
```

则以下哪一个声明不可能正确(A)

- A. `Test a;`
- B. `Test < int> a;`
- C. `Test < float> a;`
- D. `Test< Test < int> > a;`

2-7关于函数模板，描述错误的是(C)

- A.函数模板必须由程序员实例化为可执行的函数模板
- B.函数模板的实例化由编译器实现
- C.一个类定义中，只要有一个函数模板，则这个类是类模板
- D.类模板的成员函数都是函数模板，类模板实例化后，成员函数也随之实例化

2-8下列的模板说明中，正确的是(C)

A. `template < typename T1, T2 >`

B. `template < class T1, T2 >`

C. `template < typename T1, typename T2 >`

D. `template (typedef T1, typedef T2)`

2-9假设有函数模板定义如下: `template Max(T a, T b ,T &c) { c = a + b ; }` 下列选项正确的是(B)

A. `int x, y; char z ;Max(x, y, z) ;`

B. `double x, y, z ;Max(x, y, z) ;`

C. `int x, y; float z ;Max(x, y, z);`

D. `float x; double y, z;Max(x, y, z) ;`

2-10一个(C)允许用户为类定义一种模式，使得类中的某些数据成员及某些成员函数的返回值能取任意类型

A. 函数模板

B. 模板函数

C. 类模板

D. 模板类

2-11如果将运算符 * 重载为某个类的成员运算符（成员函数），则该成员函数的参数个数是(D)

A. 0个

B. 1个

C. 2个

D. 0或1个均可

2-12对定义重载函数的下列要求中，(D)是错误的

A. 要求参数的个数不同

B. 要求参数中至少有一个类型不同

C. 要求参数个数相同时，参数类型不同

D. 要求函数的返回值不同

2-13下列运算符中,(C)运算符在C++中不能被重载

A. `&&`

B. `[]`

C. `:::`

D. `new`

2-14系统在调用重载函数时往往根据一些条件确定哪个重载函数被调用，在下列选项中，不能作为依据的是(A)

A. 函数的返回值类型

B.参数的类型

C.函数名称

D.参数个数

2019FD-He练习3

2-1对于以下关于友元的说法(D)

A.如果函数fun被声明为类A的友元函数，则该函数成为A的成员函数

B.如果函数fun被声明为类A的友元函数，则该函数能访问A的保护成员，但不能访问私有成员

C.如果函数fun被声明为类A的友元函数，则fun的形参类型不能是A。

D.以上答案都不对

2-2下面关于友元的描述中，错误的是(D)

A.友元函数可以访问该类的私有数据成员

B.一个类的友元类中的成员函数都是这个类的友元函数

C.友元可以提高程序的运行效率

D.类与类之间的友元关系可以继承

2-3已知类A是类B的友元，类B是类C的友元，则(D)

A.类A一定是类C的友元

B.类C一定是类A的友元

C.类C的成员函数可以访问类B的对象的任何成员

D.类A的成员函数可以访问类B的对象的任何成员

2-4不属于类的成员函数的是(C)

A.构造函数

B.析构函数

C.友元函数

D.复制构造函数

2-5若类A被说明成类B的友元，则(D)

A.类A的成员即类B的成员

B.类B的成员即类A的成员

C.类A的成员函数不能访问类B的成员

D.类B不一定是类A的友元

2-6下列的模板说明中，正确的是(C)

A. `template < typename T1, T2 >`

B. `template < class T1, T2 >`

C. `template < typename T1, typename T2 >`

D. `template (typedef T1, typedef T2)`

2-7 假设有函数模板定义如下: `template Max(T a, T b ,T &c) { c = a + b ; }` 下列选项正确的是(B)

A. `int x, y; char z ;Max(x, y, z) ;`

B. `double x, y, z ;Max(x, y, z) ;`

C. `int x, y; float z ;Max(x, y, z) ;`

D. `float x; double y, z;Max(x, y, z) ;`

2-8 关于类模板, 描述错误的是(A)

A. 一个普通基类不能派生类模板

B. 类模板可以从普通类派生, 也可以从类模板派生

C. 根据建立对象时的实际数据类型, 编译器把类模板实例化为模板类

D. 函数的类模板参数需生成模板类并通过构造函数实例化

2-9 有函数模板定义如下:

```
template<typename T>
Max(T a, T b, T& c){c = a + b ;}
```

则下列调用中正确的是(B)

A. `int x, y; char z; Max(x, y, z);`

B. `double x, y, z; Max(x, y, z);`

C. `int x, y; float z; Max(x, y, z);`

D. `float x, double y, z; Max(x, y, z);`

2-10 下列关于模板的说法中, 错误的是(C)

A. 用模板定义一个对象时, 不能省略参数

B. 类模板只能有虚拟参数类型

C. 类模板的成员函数都是模板函数

D. 类模板在编译中不会生成任何代码

2-11 下面对模板的声明, 正确的是(C)

A. `template<T>`

B. `template<class T1, T2>`

C. `template<class T1, class T2>`

D. `template<class T1; class T2>`

2-12 下面关于继承和派生的构造函数和析构函数的程序, 输出结果是(B)

```
#include<iostream>
using namespace std;
class AA {
public:
```

```

AA() { cout << "A"; }
~AA() { cout << "a"; }
};
class BB: public AA {
    AA aa;
public:
    BB() { cout << "B"; }
    ~BB() { cout << "b"; }
};
int main() {
    BB bb;
    return 0;
}

```

A.AABaab

B.AABbaa

C.BAAaab

D.BAABaa

2-13在c++中，类之间的继承关系具有(C)

A.自反性

B.对称性

C.传递性

传递性

D.反对称性

2-14下列关于类的继承描述中，(D)是正确的

A.派生类公有继承基类时，可以访问基类的所有数据成员，调用所有成员函数。

B.派生类也是基类，所以它们是等价的。

C.派生类对象不会建立基类的私有数据成员，所以不能访问基类的私有数据成员。

D.一个基类可以有多个派生类，一个派生类可以有多个基类。

2-15下列关于运算符重载的描述中,(D)是正确的

A.运算符重载可以改变操作数的个数

B.运算符重载可以改变优先级

C.运算符重载可以改变结合性

D.运算符重载不可以改变语法结构

2-16为了能出现在赋值表达式的左右两边，重载的 [] 运算符应定义为(B)

A. A operator [] (int);

B. A& operator [] (int);

C. const A operator [] (int);

D. 以上答案都不对

2019继承上机

2-1在派生类对基类继承的传递性中, (C)是错误的

- A.在公有继承方式下, 直接派生类对象可以直接调用基类中的公有成员函数, 去访问基类的私有数据成员
- B.在公有继承方式下, 间接派生类对象可以直接调用基类中的公有成员函数, 去访问基类的私有数据成员
- C.在私有继承方式下, 间接派生类对象可以直接调用基类中的公有成员函数, 去访问基类的私有数据成员
- D.不管是私有继承还是公有继承, 基类中的私有成员在派生类的作用域内都是不可能见的。

2-2在C++语言中设置虚基类的目的是(C)

- A.简化程序代码
- B.提高程序的运行效率
- C.解决多继承造成的二义性问题
- D.缩短程序的目标代码

2-3继承机制的作用是(C)

- A.信息隐藏
- B.数据封装
- C.定义新类
- D.数据抽象

2-4C++语言类体系中, 不能被派生类继承的有(B)

- A.转换函数
- B.构造函数
- C.虚函数
- D.静态成员函数

2-5在公有继承的情况下, 在派生类中能够访问的基类成员包括(D)

- A.公有成员
- B.保护成员
- C.公有成员、保护成员和私有成员
- D.公有成员和保护成员

2-6可以用p.a的形式访问派生类对象p的基类成员a, 其中a是(D)

- A.私有继承的公有成员
- B.公有继承的私有成员
- C.公有继承的保护成员
- D.公有继承的公有成员

2-7下面关于继承和派生的构造函数和析构函数的程序, 输出结果是(B)

```
#include<iostream>
using namespace std;
class AA {
public:
    AA() { cout << "A"; }
    ~AA() { cout << "a"; }
};
class BB: public AA {
    AA aa;
public:
    BB() { cout << "B"; }
    ~BB() { cout << "b"; }
};
int main() {
    BB bb;
    return 0;
}
```

- A. AABaab
- B. AABbaa
- C. BAAaab
- D. BAABaa

2-8一个类的私有成员(B)

- A.只能被该类的成员函数访问
- B.只能被该类的成员函数和友元函数访问
- C.只能被该类的成员函数、友元函数和派生类访问
- D.以上答案都不对

2-9建立派生类对象时, 3种构造函数分别是a(基类的构造函数)、b(成员对象的构造函数)、c(派生类的构造函数), 这3种构造函数的调用顺序为(A)

- A.abc
- B.acb
- C.cab
- D.cba

2-10下面叙述不正确的是(A)

- A.基类的保护成员在派生类中仍然是保护的成员
- B.基类的保护成员在公有派生类中仍然是保护的
- C.基类的保护成员在私有派生类中仍然是私有的
- D.对基类成员的访问必须是无二义性

2-11下面关于类的继承与派生的程序, 其输出结果是(D)

```
#include<iostream>
using namespace std;
class A
```



```

{
public:
    A(){cout<<"A";}
};
class B
{
public:
    B(){cout<<"B";}
};
class C:public A
{
    B b;
public:
    C(){cout<<"C";}
};
int main(){
    C obj;
    return 0;
}

```

A.CBA

B.BAC

C.ACB

D.ABC

2-12假设在公有派生情况下，以下说法不正确的是(A)

A.可以将基类对象复制给派生类对象

B.可以将派生类对象的地址复制给基类指针

C.可以将派生类对象赋值给基类的引用

D.可以将派生类对象赋值给基类对象

2-13下列关于派生类构造函数和析构函数的说法中，错误的是(D)

A.派生类的构造函数会隐含调用基类的构造函数

B.如果基类声明了带形参表的构造函数，则派生类就应当声明构造函数

C.在建立派生类对象时，先调用基类的构造函数，再调用派生类的构造函数

D.在销毁派生类对象时，先调用基类的析构函数，再调用派生类的析构函数

2-14以下关于C++语言中继承的叙述中，错误的是(D)

A.继承是父类和子类之间共享数据和方法的机制

B.继承定义了一种类与类之间的关系

C.继承关系中的子类将拥有父类的全部属性和方法

D.继承仅仅允许单继承，即不允许一个子类有多个父类

2-15派生类继承基类的方式有(D)

A.public

B.private

C.protected

D. 以上都对

2-16下面关于类的继承与派生的程序，其输出结果是(C)

```
#include<iostream>
using namespace std;
class A {
public:
    A(int i) {    x = i;  }
    void dispa() {
        cout << x << ',';
    }
private:
    int x;
};
class B: public A {
public:
    B(int i) : A(i + 10) {
        x = i;
    }
    void dispb() {
        dispa();
        cout << x << endl;
    }
private:
    int x;
};
int main() {
    B b(2);
    b.dispb();
    return 0;
}
```

A.10,2

B.12,10

C.12,2

D.2,2

2019继承练习

2-1下列关于类的继承描述中，(D)是正确的

A.派生类公有继承基类时，可以访问基类的所有数据成员，调用所有成员函数。

B.派生类也是基类，所以它们是等价的。

C.派生类对象不会建立基类的私有数据成员，所以不能访问基类的私有数据成员。

D.一个基类可以有多个派生类，一个派生类可以有多个基类。

2-2在c++中，类之间的继承关系具有(C)

A.自反性

B.对称性

C.传递性

D.反对称性

2-3下面描述中，表达错误的是(B)

A.公用继承时基类中的public成员在派生类中仍是public的

B.公用继承时基类中的private成员在派生类中仍是private的

C.公用继承时基类中的protected成员在派生类中仍是protected的

D.私有继承时基类中的public成员在派生类中是private的

2-4以下说法正确的是(B)

A.派生类可以和基类有同名成员函数，但是不能有同名成员变量

B.派生类的成员函数中，可以调用基类的同名同参数表的成员函数

C.派生类和基类的同名成员函数必须参数表不同，否则就是重复定义

D.派生类和基类的同名成员变量存放在相同的存储空间

2-5一个类的私有成员(B)

A.只能被该类的成员函数访问

B.只能被该类的成员函数和友元函数访问

C.只能被该类的成员函数、友元函数和派生类访问

D.以上答案都不对

2-6假设在公有派生情况下，以下说法不正确的是(A)

A.可以将基类对象复制给派生类对象

B.可以将派生类对象的地址复制给基类指针

C.可以将派生类对象赋值给基类的引用

D.可以将派生类对象赋值给基类对象

2-7C++语言类体系中,不能被派生类继承的有(B)

A.转换函数

B.构造函数

C.虚函数

D.静态成员函数

2-8在下列关键字中，不能用来表示继承方式的是(C)

A.private

B.public

C.static

D.protected

2-9下列关于继承的描述中，错误的是(D)

A.析构函数不能被继承

B.派生类是基类的组合

C.派生类的成员除了它自己的成员外，还包含了它的基类的成员

D.派生类中继承的基类成员的访问权限到派生类保持不变

2-10下列关于派生类构造函数和析构函数的说法中，错误的是(D)

A.派生类的构造函数会隐含调用基类的构造函数

B.如果基类声明了带有形参表的构造函数，则派生类就应当声明构造函数

C.在建立派生类对象时，先调用基类的构造函数，再调用派生类的构造函数

D.在销毁派生类对象时，先调用基类的析构函数，再调用派生类的析构函数

2-11建立派生类对象时，3种构造函数分别是a(基类的构造函数)、b(成员对象的构造函数)、c(派生类的构造函数)，这3种构造函数的调用顺序为(A)

A.abc

B.acb

C.cab

D.cba

2-12下面关于类的继承与派生的程序，其输出结果是(C)

```
#include<iostream>
using namespace std;
class A {
public:
    A(int i) {    x = i;  }
    void dispa() {
        cout << x << ',';
    }
private:
    int x;
};
class B: public A {
public:
    B(int i) : A(i + 10) {
        x = i;
    }
    void dispb() {
        dispa();
        cout << x << endl;
    }
private:
    int x;
};
int main() {
    B b(2);
    b.dispb();
    return 0;
}
```

A.10,2

B.12,10

C.12,2

D.2,2

2019多态上机

2-1下面的概念，哪个不是关于对象的多态性的体现**(B)**

A.方法的重载

B.方法的继承

C.方法的覆盖

D.对象的上、下转型

2-2关于面向对象的特征**(D)**不是面向对象的特征

A.多态性

B.继承性

C.封装性

D.过程调试

2-3在C++中用类将数据和对数据操作的代码连接在一起称为**(B)**

A.软件重用

B.封装

C.集合

D.多态

2-4在派生类中，重载一个虚函数时，要求函数名、参数的个数、参数的类型、参数的顺序和函数的返回值**(A)**

A.相同

B.不同

C.相容

D.部分相同

2-5虚析构函数的作用是**(C)**

A.虚基类必须定义虚析构函数

B.类对象作用域结束时释放资源

C.delete动态对象时释放资源

D.无意义

2-6下面函数原型中，**(C)**声明了fun为纯虚函数

A.void fun()=0

B.virtual void fun()=0

C.virtual void fun()

D.virtual void fun(){ }

2-7若一个类中含有纯虚函数，则该类称为(D)

- A.基类
- B.纯基类
- C.抽象类
- D.派生类

2-8下面描述中，正确的是(D)

- A.虚函数是没有实现的函数
- B.纯虚函数是返回值等于0的函数
- C.抽象类是只有纯虚函数的类
- D.抽象类指针可以指向不同的派生类

2019多态练习

2-1虚析构函数的作用是(C)

- A.虚基类必须定义虚析构函数
- B.类对象作用域结束时释放资源
- C.delete动态对象时释放资源
- D.无意义

2-2若一个类中含有纯虚函数，则该类称为(C)

- A.基类
- B.纯基类
- C.抽象类
- D.派生类

2-3以下说法中正确的是(B)

- A.在虚函数中不能使用this指针
- B.在构造函数中调用虚函数不是动态联编
- C.抽象类的成员函数都是纯虚函数
- D>构造函数和析构函数都不能是虚函数

2-4关于虚函数的描述中,(C)是正确的

- A.虚函数是一个 static 类型的成员函数
- B.虚函数是一个非成员函数
- C.基类中说明了虚函数后，派生类中与其对应的函数可不说明为虚函数
- D.派生类的虚函数与基类的虚函数具有不同的参数个数和类型

2-5关于纯虚函数和抽象类的描述中,(B)是错误的

- A.纯虚函数是一种特殊的虚函数，它没有具体的实现
- B.一个基类中说明有纯虚函数，该基类的派生类一定不再是抽象类

C.抽象类是指具有纯虚函数的类

D.抽象类只能作为基类来使用，其纯虚函数的实现由派生类给出

2-6下面描述中，正确的是(D)

A.虚函数是没有实现的函数

B.纯虚函数是返回值等于0的函数

C.抽象类是只有纯虚函数的类

D.抽象类指针可以指向不同的派生类

2-7关于动态绑定的下列描述中，(D)是错误的

A.动态绑定是以虚函数为基础的

B.动态绑定在运行时确定所调用的函数代码

C.动态绑定调用函数操作是通过指向对象的指针或对象引用来实现的

D.动态绑定是在编译时确定操作函数的

2-8下列描述中，(A)是抽象类的特性

A.可以说明虚函数

B.可以进行构造函数重载

C.可以定义友元函数

D.不能定义该类对象

2-11设有如下代码段:

```
class A {
public:
    void func1() {
        cout << "A1" << endl;
    }
    virtual void func2() {
        cout << "A2" << endl;
    }
};

class B : public A {
public:
    void func1() {
        cout << "B1" << endl;
    }
    void func2() {
        cout << "B2" << endl;
    }
};

int main() {
    A *a = new B;
    a->func1();
    a->func2();
}
```

则输出结果为(B)

A.A1 A2

B.A1 B2

C.B1 A2

D.B1 B2

ABC