**C++知识点整理**

1. ***：***
2. **对象**：对象是由一些数据及可施于这些数据上的操作所组成的封装体。
3. **语言分类**：与自然语言的接近程度

（1）低级语言：机器语言、汇编语言

（2）高级语言：程序设计语言

1. **程序运行步骤**：编辑——编译——连接——运行
2. **程序设计步骤**：需求分析、系统设计、编程实现、测试与调试、运行与维护
3. **标识符**：（比如变量名称）
4. **关键字**：注意大小写及拼写（e.g.external、If、integer）
5. ***：***
6. **C++基本数据类型（标准类型、内置类型）：**
7. **整数类型**：*short int*：2个字节；*int/long int*：4个字节；*long long int*：8个字节；

（2）**实数类型：***float*：4个字节；*double/long double*：8个字节；

（3）**字符类型**：*char*：1个字节；（string不是基本数据类型）

（4）**逻辑类型**：*bool*：1个字节；

（5）**空值类型**：*void*：0个字节（void\*通用指针类型）

1. **数据类型：简单数据类型、构造数据类型、抽象数据类型**

（1）简单数据类型：整数、实数、字符、逻辑、空值

（2）构造数据类型：指针、引用、数组、结构体、枚举、联合

（3）抽象数据类型：类、派生类

1. **常量：字面常量、符号常量**
2. 字面常量：整数类型、实数类型、字符类型、字符串

（2）符号常量

1. **操作符**：
2. 算数操作符：+、-、\*、/、%、++、--
3. 关系操作符：>、<、>=、<=、==、!=
4. 逻辑操作符：&&、｜｜、! （注意短路求值）
5. 逻辑位操作符：～（按位取反）、&（按位取与）、｜（按位取或）、^（按位异或）（x^a）^a=x、<<、>>
6. 赋值操作符：（简单）=、（复合）+=、-=、\*=、/=、%=、&=、｜=、^=、<<=、>>=
7. 其他操作符：?: 、 ， 、sizeof
8. **类型转换**：
9. 隐式类型转换
10. 显式类型转换
11. **优先级与结合性**：
12. 优先级：决定表达式中各种不同的运算符起作用的优先次序
13. 结合性：在相邻的两个运算符的具有同等优先级时，决定表达式的结合方向
14. ***：***
15. **语句**
16. 顺序执行语句：表达式语句、复合语句、空语句
17. 选择执行语句：if语句、switch语句
18. 循环执行语句：while语句、while-do语句、for语句
19. 无条件转移语句：goto语句、break语句、continue语句、return语句
20. **C++内部不能定义局部常量**
21. ***：***
22. **子程序间的数据传递**
23. 值传递
24. 地址传递

（3）引用传递

1. **参数传递**
2. 值传递
3. 地址传递

（3）引用传递

1. **全局变量与局部变量**
2. **命名空间：在一个命名空间中定义的全局标识符，其作用域为该命名空间**
3. **作用域与生存期**
4. 作用域：局部作用域、全局作用域、文件作用域、函数作用域、函数原型作用域、命名空间作用域、类作用域
5. 生存期：静态生存期、自动生存期、动态生存期
6. static：全局变量改为文件作用域、局部变量具有静态生存期
7. auto：局部变量默认、形参
8. register：使用频率高的局部变量
9. 静态生存期：static局部变量、全局变量
10. 自动生存期：auto局部变量、函数形参、register
11. 动态生存期：new、delete、malloc、free
12. 内存空间：静态数据区、代码区、栈区、堆区
13. 静态数据区（大小固定）：全局变量、static局部变量、某些常量
14. 代码区（大小固定）：函数代码
15. 栈区：auto局部变量、函数形参、函数调用有关信息
16. 堆区：动态变量内存分配
17. **内联函数**
18. 内联函数：在函数定义中的返回值类型之前加上关键字inline
19. 作用：建议编译程序把该函数的函数体展开到调用点，避免函数调用的开销，提高函数调用的效率
20. ***：***
21. **数组类型（排序）**
22. 定义：一种由固定多个同类型的具有一定次序关系的元素所构成的复合数据类型
23. 排序：
24. 冒泡排序法：

void BubbleSort(int arr[], int n)

{

for (**int i = 0; i < n - 1;** i++)

{

for (**int j = 0; j < n - i - 1;** j++)

{

if (arr[j] > arr[j + 1])

{

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

1. 选择排序法：

void selectSort(int a[], int len)

{

int minindex, temp;

for(**int i = 0; i<len-1**;i++)

{

minindex = i;

for(**int j = i+1; j<len;** j++)

{

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

minindex = j;

}

temp = a[i];

a[i] = a[minindex];

a[minindex] = temp;

}

}

1. 快速排序法：

void quickSort(int a[], int low ,int high)

{

if(low<high) //判断是否满足排序条件，递归的终止条件

{

int i = low, j = high;

int x = a[low];

//将待排序数组的第一个元素作为哨兵，将数组划分为大于哨兵以及小于哨兵的两部分

while(i<j)

{

while(i<j && a[j] >= x) j--;

//从最右侧元素开始，如果比哨兵大，那么它的位置就正确，然后判断前一个元素，直到不满足条件

if(i<j) a[i++] = a[j];

//把不满足位次条件的那个元素值赋值给第一个元素，（也即是哨兵元素，此时哨兵已经保存在x中，不会丢失）并把i的加1

while(i<j && a[i] <= x) i++;

//换成左侧下标为i的元素开始与哨兵比较大小，比其小，那么它所处的位置就正确，然后判断后一个，直到不满足条件

if(i<j) a[j--] = a[i];

//把不满足位次条件的那个元素值赋值给下标为j的元素，（下标为j的元素已经保存到前面，不会丢失）并把j的加1

}

a[i] = x;

//完成一次排序，把哨兵赋值到下标为i的位置，即前面的都比它小，后面的都比它大

quickSort(a, low ,i-1);

//递归进行哨兵前后两部分元素排序，low,high的值不发生变化，i处于中间

quickSort(a, i+1 ,high);

}

1. **枚举类型（enum）**
2. **结构体（struct）**
3. **联合类型（union）**
4. **指针类型**
5. 定义：指针是内存地址的抽象表示
6. （void\*）输出字符串内存首地址
7. const
8. const int\* p：指向常量的指针（不能改变元素的值）
9. int\* const p：指针类型的常量（可以改变所指变量的值，但不能改变指针本身的值）
10. const 成员函数只能获取对象的状态，而不能改变对象的状态
11. 交换参数
12. 多级指针

void swap(**int \*\*x,int \*\*y**)

{

int **\*t**;

t=\*x;

\*x=\*y;

\*y=t;

}

1. 指针加引用

void swap(**int \*&x,int \*&y**)

{

int **\*t**;

t=x;x=y;y=t;

}

1. 引用

void swap(**int &x,int &y**)

{

int **t**;

t=x;x=y;y=t;

}

1. 指针

void swap (**int \*x, int \*y**)

{ int **t**;

t = \*x;

\*x = \*y;

\*y = t;

}

f. **链表**

g. 引用类型

1. 作为参数：相当于别名，可提高参数传递效率
2. 引用：直接访问，参数是一个变量的名字

指针：间接访问，参数是一个变量的地址

1. 不能再引用其他变量
2. ***：***
3. **面向对象**
4. 面向对象程序设计：把程序构造成由若干对象组成，每个对象由一些数据以及对这些数据所能实施的操作构成；对对象的操作是通过向包含数据的对象发消息来实现的；对象的特征由相应的类来描述，一个类所描述的对象特征可以从其他类继承；
5. 对象：对象构成了面向对象程序的基本计算单位，它由数据、操作以及对外接口构成；
6. 通信：通信是指对象间的消息传递，它是引起面向对象程序进行计算的唯一方式；
7. 类：类描述了一组具有相同特征的对象；
8. 类的成员：**数据成员**和**成员函数**；
9. 继承：继承是指对象的一部分特征描述可以从其他的类获得；（派生类&基类）
10. 对象和类体现了**数据抽象**和**封装**的概念。类是对象的集合；对象是类的实例。
11. 多态性：某一论域中的一个元素存在多种解释；不同的对象调用相同名称的函数。
12. 绑定：指对多态元素确定使用的过程，即确定对多态元素的某个使用是多态元素的哪一种形式；（静态绑定&动态绑定）
13. 类属性：指一个程序实体能对多种类型的数据进行操作或描述的特性。通过**指针**和**函数模板**实现类属函数；通过**联合类型**和**类模板**实现类属类型。

b. **数据成员**

1. 数据成员初始化：赋值函数、构造函数、成员初始化表
2. 信息隐藏：public、private、protected

c. **对象**

1. 创建方式：直接方式、间接方式
2. 直接方式：A a；
3. 间接方式：A \*p; p=new A; delete p;
4. 如果变量的类型为类，则称变量为对象；
5. this指针

d. **构造函数**

1. 定义：在对象类中定义或声明的与类同名、无返回值类型的成员函数；
2. 在创建对象时，构造函数自动调用；创建之后不能再调用；
3. **调用顺序：基类->对象->派生类**
4. **构造函数不能被派生类继承**

e. **拷贝构造函数**

1. 被调用情况：
2. 用类的一个已知的对象去初始化该类的另一个对象时；
3. 函数的形参是类的对象，调用函数进行形参和实参的结合时；
4. 函数的返回值是类的对象，函数执行完返回调用者。

（2）调用顺序

f. **转移构造函数**

1. 参数类型：右值引用类型，要求只能是临时对象或要消亡的对象
2. 意义：用即将消亡的对象去初始化另一个同类的对象
3. 函数：

A(A&& x)

{

p=x.p;

x.p=NULL;

}

g. **析构函数**

1. 在对象消亡前，在系统收回它所占的内存空间之前，会自动调用对象类的析构函数。
2. 数组：delete **[]**p; （[] 必须写，否则只调用数组第一个对象的析构函数）
3. **调用顺序：派生类->对象->基类**

h. **const**

（1）若对象为const类，则调用函数必为const函数

i. **静态数据成员**

1. 类的静态成员：静态数据成员、静态成员函数
2. 静态数据成员往往在类的外部给出定义并进行初始化。
3. **类定义中的静态数据成员对于该类的所有对象只存在一个拷贝**。
4. **静态成员函数只能访问静态成员**；
5. **静态成员函数没有隐藏的this指针**（因为静态成员函数对静态数据成员操作，静态数据成员只有一个拷贝）
6. 私有的

j. **友元**

1. 在C++的一个类定义中，可以指定某个全局函数、某个其他类或某个其他类的某个成员函数能直接访问该类的私有和保护成员，分别称为：友元函数、友元类、友元类成员函数，统称友元。
2. 作用：提高对象私有数据成员的访问效率

k. **操作符重载（转移构造函数、友元的使用）**

1. ***：***
2. ***：***
3. ***：***
4. ***：***