C++ overview:

1. Overload: Note that a function cannot be overloaded only by its return type. At least one of its parameters must have a different type.
2. Template: template <template-parameters> function-declaration

|  |  |
| --- | --- |
| template <class SomeType> SomeType sum (SomeType a, SomeType b) { return a+b; } |  |

1. Instantiating a template: name <template-arguments> (function-arguments)

|  |  |
| --- | --- |
| x = sum<int>(10,20); |  |

1. Non type template arguments

|  |  |
| --- | --- |
| // template arguments #include <iostream> using namespace std; template <class T, int N> T fixed\_multiply (T val) { return val \* N; } int main() { std::cout << fixed\_multiply<int,2>(10) << '\n'; std::cout << fixed\_multiply<int,3>(10) << '\n'; } |  |

1. Class : Uniform initialization
   1. First, constructors with a single parameter can be called using the variable initialization syntax (an equal sign followed by the argument):

|  |  |
| --- | --- |
| class\_name object\_name = initialization\_value;  class\_name object\_name { value, value, value, ... }  int main () {  Circle foo (10.0); // functional form  Circle bar = 20.0; // assignment init.  Circle baz {30.0}; // uniform init.  Circle qux = {40.0}; // POD-like |  |

1. Memer initialization in constructors

|  |  |
| --- | --- |
| Rectangle::Rectangle (int x, int y) : width(x) { height=y; } |  |

|  |  |
| --- | --- |
| Rectangle::Rectangle (int x, int y) : width(x), height(y) { } |  |

1. **const**objects only have access to const member functions.
2. Member functions can be **overloaded** on their **constness**: i.e., a class may have two member functions with identical signatures except that one is const and the other is not: in this case, the const version is called only when the object is itself const, and the non-const version is called when the object is itself non-const
3. **Class templates**

|  |  |
| --- | --- |
| template <class T> class mypair { T values [2]; public: mypair (T first, T second) { values[0]=first; values[1]=second; } }; |  |
| mypair<int> myobject (115, 36); |  |

1. The constructor is the only member function in the previous class template and it has been defined inline within the class definition itself. In case that a member function is defined outside the defintion of the class template, it shall be preceded with the template <...> prefix:

|  |  |
| --- | --- |
| template <class T> T mypair<T>::getmax () |  |

1. **Template specialization**

|  |  |
| --- | --- |
| // template specialization  #include <iostream>  using namespace std;    // class template:  template <class T>  class mycontainer {  T element;  public:  mycontainer (T arg) {element=arg;}  T increase () {return ++element;} };  // class template specialization:  template <> class mycontainer <char> {  char element;  public:  mycontainer (char arg) {element=arg;}  char uppercase () {  if ((element>='a')&&(element<='z'))  element+='A'-'a';  return element;  }  };  int main () {  mycontainer<int> myint (7);  mycontainer<char> mychar ('j');  cout << myint.increase() << endl;  cout << mychar.uppercase() << endl;  return 0; } |  |

TCPDump 教程

1. -n 不解析域名， 用IP形式显示地址。 -nn 不需要解析主机名和端口名
2. -I any 监听所有界面
3. -X 把包的内容咦ASCII和16制形式显示。 -XX： 同-X，连Ethernet包头也显示
4. -S 显示绝对序号
5. tcpdump默认只取出前96个字节 否则用-s \*\*\*\*（总字节数）
6. -v， -vv, -vvv 详细详细再详细
7. -e 得到Ethernet头部
8. -c抓取x个包就停下
9. -q 显示少一点协议信息
10. -s: 设置显示前多少个字节的包内容（snaplength）
11. -s 1514 得到包的所有内容
12. 表达式：type, dir, proto. Type: src & dst
    1. host:只查看指定主机名 #tcpdump host 192.168.1.2
    2. src/dst:查看从src发出或者到dst的流量 #tcpdump src 1.2.3.4 #tcpdump dst 1.2.3.4
    3. net:使用无类别域间路由指定要抓包的网络 #tcpdump net 1.2.3.0/24
    4. proto：指定协议类型。 tcp/udp/icmp 可以不用输入proto #tcpdump icmp
    5. port：查看从指定端口进出的流量 #tcpdump port 3389
    6. src, dst port: 指定的源地址和端口号，或者目的地址和端口号。只抓取匹配指定地址端口的流量
       1. tcpdump src port 12345
       2. tcpdump dst port 23456
       3. tcpdump src port 12345 and tcp
       4. tcpdump udp and src port 53
    7. port range指定端口范围： #tcpdump portrange 21-23
    8. packet size指定包的大小(in bytes)：
       1. #tcpdump less 32 或者 tcpdump <= 32
       2. #tcpdump greater 128 或者tcpdump > 128
       3. #tcpdump equal 64
    9. -w 写入文件，将抓到的内容写入该文件。-r指定文件，将预先存入的内容读回来
       1. 把出入80端口的所有流量记录到文件中： #tcpdump -s 1514 port 80 -w file01
       2. tcpdump -r file01
    10. tcpdump的组合操作：与或非 # and &, or ||, not !
        1. 检测来自1.2.3.4去端口1234的流量： #tcpdump -nnvS tcp and src 1.2.3.4 and dst port 1234
        2. 来自网络 172.16.16.0/16 去到主机 192.168.0.2的非ICMP 流量: #tcpdump -nnvXSs 1514 dst 192.168.0.2 and src net 172.16.16.0/16 and not icmp
    11. 用单引号忽略特殊字符： #tcpdump ‘src 10.0.2.4 and (dst port 3389 or 22)’
    12. 根据字段指定条件：
        1. 查看紧急包（URG）： #tcpdump ‘tcp[13] & 32 != 0’
        2. 查看所有的确认包(ACK包) # tcpdump ‘tcp[13] & 16 != 0′
        3. 查看所有的PSH包 # tcpdump ‘tcp[13] & 8 != 0′
        4. 查看所有的RST包 # tcpdump ‘tcp[13] & 4 != 0′
        5. 查看所有的SYN包 # tcpdump ‘tcp[13] & 2 != 0′
        6. 查看所有的FIN包 # tcpdump ‘tcp[13] & 1 != 0′
        7. 查看所有的SYN-ACK包 # tcpdump ‘tcp[13] = 18′