University of Hertfordshire

7COM1025
Programming for Software Engineers
Lecture 24

Dr. Renato C. de Amorim

THE ONE HUNDRED UNDER FIFTY

NESTED STRUCTS

```
#include <iostream>
#include <string>
using namespace std;
struct Address {
  string StreetAddress="";
  string PostCode="";
  string Country="";
  string GetFullAddress()
  {return StreetAddress + " " + PostCode + " " + Country;}};
struct Person{
    string Name;
    Address MyAddress;
    string GetDetails(){return Name + " " + MyAddress.GetFullAddress();}};
int main (){
    Person aPerson;
    aPerson.Name = "John Smith";
    aPerson.MyAddress.StreetAddress="College Lane";
    aPerson.MyAddress.PostCode="AL10 9AB";
    aPerson.MyAddress.Country="UK";
    cout<<aPerson.GetDetails();
    return 0;}
```

POINTERS TO STRUCTS

```
#include <iostream>
#include <string>
using namespace std;
struct Address {
  string StreetAddress="";
  string PostCode="";
  string Country="";
  string GetFullAddress()
  {return StreetAddress + " " + PostCode + " " + Country;}};
int main (){
    Address *myAddress = new Address;
    myAddress->StreetAddress = "College Lane";
    myAddress->PostCode="AL10 9AB";
    myAddress->Country="UK";
    cout<<myAddress->GetFullAddress()<<endl;
    delete myAddress;
    return 0;}
```

POINTERS AND NESTED STRUCTS

```
#include <iostream>
#include <string>
using namespace std;
struct Address {
  string StreetAddress="";
  string PostCode="";
  string Country="";
  string GetFullAddress()
  {return StreetAddress + " " + PostCode + " " + Country;}};
struct Person{
    string Name;
    Address *MyAddress = new Address;
    string GetDetails(){return Name + " " + MyAddress->GetFullAddress();}
    ~Person(){delete MyAddress;}};
int main (){
    Person *aPerson = new Person;
    aPerson->Name = "John Smith";
    aPerson->MyAddress->StreetAddress="College Lane";
    aPerson->MyAddress->PostCode="AL10 9AB";
    aPerson->MyAddress->Country="UK";
    cout<<aPerson->GetDetails();
    delete aPerson;
    return 0;}
```

PROBLEM 24.1

Create a data structure capable of keeping information about computers. Each computer has an IP address, value for RAM (GB or MB), Hard disk and CPU. The CPU has a Name, it's either 32 or 64 bits, a speed (either in GHz or MHz).

Write a program capable of storing as many CPUs as the user wishes.

STL EXCEPTION CLASS

Example of bad allocation

```
#include <iostream>
#include <exception>
using namespace std;
int main () {
  try {
    double* myarray= new double[9999999999999999]; }
  catch (exception& e) {
    cout << "Standard exception: " << e.what() << endl; }
  return 0;
}
```

STL EXCEPTION CLASS

It has a virtual destructor and “what” is a virtual function!

```
#include <iostream>
#include <exception>
using namespace std;
class myexception: public exception {
  virtual const char* what() const throw() {
    return "Exception thrown"; }
} myex;
int main () {
  try {
    throw myex; }
  catch (exception& e) {
    cout << e.what() << '\n';
    }
  return 0;
}
```

STL EXCEPTION CLASS

It has a virtual destructor and “what” is a virtual function!

```
#include <iostream>
#include <exception>
using namespace std;
class myexception: public exception {
  virtual const char* what() const throw() {
    return "Exception thrown"; }
} myex;
int main () {
  try {
    throw myex; }
  catch (exception& e) {
    cout << e.what() << '\n';
    }
  return 0;
}
```

PROBLEM 24.2

Remember your exception class?

Updated so that it inherits from the STL.

Now update your sorting program so that it throws an exception if there is no enough memory, or if the user does not enter a number when requested.