MA122 - Computer Programming and Apllications

Indian Institute of Space Science and Technology

February 23, 2017

Lecture 16

1 pointers and const

2 2D array

3 Using Structures with functions

4 Recursion

pointers and const


```
1  //intcont.cpp
2  #include <iostream>
3  int main()
4  {
5    int x=16;
6    int y=20;
7
8    const int *pt1;
9    pt1=&y;
10 //  *pt1=22;
11
12   int* const pt2;
13
14   *pt2=55;
15   //pt2=&x;
16
17   return 0;
18 }
```

Lecture 16

1 pointers and const

2 2D array

3 Using Structures with functions

4 Recursion

1. int data[3][4]: data is pointer-to-array-of-four-int
2. appropriate prototype: int sum(int (*ar2)[4], int size);
3. same as: int sum(int ar2[][4], int size);
4. int *ar2[4]: four pointers-to-int instead of a single pointer-to-array-of-four-int
5. ar2[r][c] == *(*(ar2 + r) + c)

```
ar2                 // pointer to first row of an array of 4 int
ar2 + r             // pointer to row r (an array of 4 int)
*(ar2 + r)          // row r (an array of 4 int, hence the name of an array,
                    // thus a pointer to the first int in the row, i.e., ar2[r]

*(ar2 +r) + c       // pointer int number c in row r, i.e., ar2[r] + c
*(*(ar2 + r) + c // value of int number c in row r, i.e. ar2[r][c]
```

2D array


```
1  //2darray.cpp
2  #include <iostream>
3
4  using namespace std;
5  int sum(int (*ar2)[4],int size);
6  int main()
7  {
8    int data[3][4]={{1,2,3,4},{9,8,7,6},{12,14,16,18}};
9
10
11   cout<<endl;
12
13     cout<<(*data)<<" "<<(*data)+1<<" "<<*(data)+2
14   <<"  "<<*(data)+3<<endl<<endl;
15
16     cout<<*(data+1)<<" "<<*(data+1)+1<<" "
17   <<*(data+1)+2<<" "<<*(data+1)+3<<endl<<endl;
```

2D array


```
1     cout<<*(data+2)<<" "<<*(data+2)+1<<" "
2 <<*(data+2)+2<<" "<<*(data+2)+3<<endl<<endl;
3
4
5
6     cout<<*((*data))<<" "<<*((*data)+1)<<" "
7 <<*(*(data)+2)<<" "<<*(*(data)+3)<<endl<<endl;
8
9
10
11    cout<<*(*(data+1))<<" "<<*(*(data+1)+1)<<" "
12 <<*(*(data+1)+2)<<" "<<*(*(data+1)+3)<<endl<<endl;
13
14
15    cout<<*(*(data+2))<<" "<<*(*(data+2)+1)<<" "
16 <<*(*(data+2)+2)<<" "<<*(*(data+2)+3)<<endl;
```

2D array


```
1
2       cout<<endl<<"sum= "<<sum(data,4)<<endl;
3
4     return 0;
5   }
6
7   int sum(int (*ar2)[4], int size)
8   {
9       int sum=0.0;
10      for(int i=0;i<3;i++)
11      {
12          for(int j=0;j<size;j++)
13          {
14              sum=sum+ar2[i][j];
15          }
16      }
17      return sum;
18  }
```

Lecture 16

1 pointers and const

2 2D array

3 Using Structures with functions

4 Recursion

Structures with functions


```
1  //structureswithfunctions.cpp
2  #include <iostream>
3  struct travel_time
4  {
5    int hours;
6    int mins;
7  };
8
9  const int Mins_per_hr = 60;
10
11 travel_time sum(travel_time t1, travel_time t2);
12
13 void show_time(travel_time t);
14
15 int main()
16 {
17 using namespace std;
```

Structures with functions


```
1
2     travel_time day1 = {5, 45};
3
4     travel_time day2 = {4, 55};
5
6     travel_time trip = sum(day1, day2);
7
8     cout << "Two-day total: ";
9     show_time(trip);
10
11    travel_time day3= {4, 32};
12    cout << "Three-day total: ";
13
14    show_time(sum(trip, day3));
15
16    return 0;
17 }
```

Structures with functions


```
1  travel_time sum(travel_time t1, travel_time t2)
2  {
3    travel_time total;
4
5    total.mins = (t1.mins + t2.mins) % Mins_per_hr;
6
7    total.hours = t1.hours + t2.hours +
8    (t1.mins + t2.mins) / Mins_per_hr;
9
10   return total;
11 }
12 void show_time(travel_time t)
13 {
14   using namespace std;
15
16   cout << t.hours << " hours, "
17   << t.mins << " minutes\n";
18 }
```

Lecture 16

1 pointers and const

2 2D array

3 Using Structures with functions

4 Recursion

single recursive call


```
1  // recursivefunction.cpp -- using recursion
2  #include <iostream>
3  void countdown(int n);
4  int main()
5  {
6    countdown(4);          // call the recursive function
7    return 0;
8  }
9  void countdown(int n)
10 {
11   std::cout << "Counting down ... " << n <<" "<<&n<<
         std::endl;
12
13   if (n > 0)
14     countdown(n-1);    // function calls itself
15   std::cout << n << ": back out!\n";
16 }
```

Multiple recursion


```
1  //multiplerecursion.cpp
2  #include <iostream>
3  const int Len = 66;
4  const int Divs = 6;
5  void subdivide(char ar[], int low, int high, int level
       );
6
7  int main()
8  {
9    char ruler[Len];
10   int i;
11
12   for (i = 1; i < Len - 2; i++)
13     ruler[i] = ' ';
14
15   ruler[Len - 1] = '\0';
16   int max = Len - 2;
17   int min = 0;
```

Multiple recursion


```
1
2     ruler[min] = ruler[max] = '|';
3
4     std::cout << ruler << std::endl;
5
6     for (i = 1; i <= Divs; i++)
7     {
8       subdivide(ruler,min,max, i);
9
10      std::cout << ruler << std::endl;
11
12      for (int j = 1; j < Len - 2; j++)
13        ruler[j] = ' ';
14    }
15    return 0;
16  }
```

Multiple recursion


```
1  void subdivide(char ar[], int low, int high, int level
       )
2  {
3
4    if (level == 0)
5      return;
6
7    int mid = (high + low) / 2;
8
9    ar[mid] = '|';
10
11   subdivide(ar, low, mid, level - 1);
12   subdivide(ar, mid, high, level - 1);
13 }
```