MA122 - Computer Programming and Apllications

Indian Institute of Space Science and Technology

March 10, 2017

Lecture 20

Initialization


```
1  #include <iostream>
2  using namespace std;
3
4  int main () {
5      double* pvalue = NULL; // Pointer initialized with
           null
6      pvalue  = new double;  // Request memory for the
           variable
7
8      *pvalue = 29494.1;   // Store value at allocated
           address
9      cout << "Value of pvalue : " << *pvalue << endl;
10     delete pvalue;         // free up the memory.
11
12     return 0;
13 }
```

Initialization


```
1  #include <iostream>
2  int main()
3  {
4    using namespace std;
5    int nights = 1001;
6
7    int * pt = new int;       // allocate space for an
         int
8    *pt = nights;               // store a value there
9
10   cout << "nights value = ";
11   cout << nights << ": location " << &nights << endl;
12
13   cout << "int ";
14     cout << "value = " << *pt << ": location = " << pt
           << endl;
```

Initialization


```
1   double * pd = new double; // allocate space for a
        double
2   *pd = 10000001.0;  // store a double there
3
4   cout << "double ";
5   cout << "value= " << *pd << ": location = " << pd <<
        endl;
6
7   cout << "location of pointer pd: " << &pd << endl;
8   cout << "size of pt = " << sizeof(pt);
9
10  cout << ": size of *pt = " << sizeof(*pt) << endl;
11  cout << "size of pd = " << sizeof pd;
12
13  cout << ": size of *pd = " << sizeof(*pd) << endl;
14  return 0;
15 }
```

array


```
1  #include <iostream>
2  int main()
3  {
4    using namespace std;
5    double * p3 = new double [3]; // space for 3 doubles
6    p3[0] = 0.2;                  // treat p3 like an array
         name
7    p3[1] = 0.5;
8    p3[2] = 0.8;
9    cout << "p3[1] is " << p3[1] << ".\n";
10   p3 = p3 + 1;                  // increment the pointer
11   cout << "Now p3[0] is " << p3[0] << " and ";
12   cout << "p3[1] is " << p3[1] << ".\n";
13   p3 = p3 - 1;                  // point back to
       beginning
14   delete [] p3;                 // free the memory
15   return 0;
16 }
```

Initialization


```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  int main()
6  {
7      int num;
8      cout << "Enter total number of students: ";
9      cin >> num;
10     float* ptr;
11
12     // memory allocation of num number of floats
13     ptr = new float[num];
```

Initialization


```
    cout << "Enter GPA of students." << endl;
    for (int i = 0; i < num; ++i)
    {
        cout << "Student" << i + 1 << ": ";
        cin >> *(ptr + i);
    }

    cout << "\nDisplaying GPA of students." << endl;
    for (int i = 0; i < num; ++i) {
        cout << "Student" << i + 1 << " :" << *(ptr +
            i) << endl;
    }

    // ptr memory is released
    delete [] ptr;

    return 0;
}
```

array


```
1  #include <iostream>
2  #include <new>
3  using namespace std;
4
5  int main ()
6  {
7    int i,n;
8    int * p;
9    cout << "How many numbers would you like to type? ";
10   cin >> i;
11   p= new int[i];
12
13     for (n=0; n<i; n++)
14     {
15       cout << "Enter number: ";
16       cin >> p[n];
17     }
```

array


```
1
2     cout << "You have entered: ";
3     for (n=0; n<i; n++)
4       cout << p[n] << ", ";
5     delete[] p;
6
7   return 0;}
```

Lecture 20

example


```
1  // overloading functions
2  #include <iostream>
3  using namespace std;
4  int operate(int, int);
5  double operate(double,double);
6  int main ()
7  {
8      int x=5,y=2;
9      double n=5.0,m=2.0;
10     cout << operate (x,y) << '\n';
11     cout << operate (n,m) << '\n';
12     return 0;
13 }
```

example


```
1
2  int operate (int a, int b)
3  {
4      return (a*b);
5  }
6
7  double operate (double a, double b)
8  {
9      return (a/b);
10 }
```

```
void staff(double & rs);          // matches modifiable lvalue
voit staff(const double & rcs);   // matches rvalue, const lvalue
void stove(double & r1);          // matches modifiable lvalue
void stove(const double & r2);    // matches const lvalue
void stove(double && r3);         // matches rvalue
```

This allows you to customize the behavior of a function based on the lvalue, `const`, or rvalue nature of the argument:

```
double x = 55.5;
const double y = 32.0;
stove(x);              // calls stove(double &)
stove(y);              // calls stove(const double &)
stove(x+y);            // calls stove(double &&)
```

If, say, you omit the `stove(double &&)` function, then `stove(x+y)` will call the `stove(const double &)` function instead.

example


```
1  #include <iostream>
2  unsigned long left(unsigned long num, unsigned ct);
3  char * left(const char * str, int n = 1);
4  int main()
5  {
6    using namespace std;
7    char * trip = "Hawaii!!"; // test value
8    unsigned long n = 12345678; // test value
9    int i;
10   char * temp;
11   for (i = 1; i < 10; i++)
12   {
13     cout << left(n, i) << endl;
14     temp = left(trip,i);
15     cout << temp << endl;
16     delete [] temp; // point to temporary storage
17   }
18 return 0;
```

example


```
1  unsigned long left(unsigned long num, unsigned ct)
2  {
3    unsigneddigits = 1;
4    unsigned long n = num;
5    if (ct == 0 || num == 0)
6      return 0;       // return 0 if no digits
7    while (n /= 10)
8      digits++;
9    if (digits > ct)
10   {
11     ct = digits - ct;
12     while (ct--)
13       num /= 10;
14     return num;         // return left ct digits
15   }
16   else                  // if ct >= number of digits
17     return num;     // return the whole number
18 }
```

example


```
1
2  char * left(const char * str, int n)
3  {
4    if(n < 0)
5      n = 0;
6    char* p = new char[n+1];
7    int i;
8    for (i = 0; i < n && str[i]; i++)
9      p[i] = str[i]; // copy characters
10   while (i <= n)
11     p[i++] = '\0'; // set rest of string to '\0'
12   return p;
13 }
```