Data Structures and Object Oriented Programming using C++

Ahsan Ijaz

September 16, 2013

Topics Covered

- Basic Data Types
- Variables, Constants
- Functions, Function Overloading
- Pointers
- Arrays
- Dynamic Memory Allocation

Topics Covered

- Basic Data Types
- Variables, Constants
- Functions, Function Overloading
- Pointers
- Arrays
- Dynamic Memory Allocation

Topics Covered

- Basic Data Types
- Variables, Constants
- Functions, Function Overloading
- Pointers
- Arrays
- Dynamic Memory Allocation

Topics Covered

- Basic Data Types
- Variables, Constants
- Functions, Function Overloading
- Pointers
- Arrays
- Dynamic Memory Allocation

Topics Covered

- Basic Data Types
- Variables, Constants
- Functions, Function Overloading
- Pointers
- Arrays
- Dynamic Memory Allocation

Topics Covered

- Basic Data Types
- Variables, Constants
- Functions, Function Overloading
- Pointers
- Arrays
- Dynamic Memory Allocation

Basic Program

```
1
2 #include <iostream>
3 using namespace std;
4 /*Loads of Comments*/
5 int main ()
6 {
7    cout << "Hello World!"; // But of course!
8    return 0;
9 }
```

Data Types

Name	Description	Size (byte)
char	Character	1
short int (short)	Short integer	2
int	Integer	4
long int (long)	Long integer	4
bool	Boolean value (1 or 0)	1
float	Floating point variable	4
double	Double precision floating point	8

Table: Data Types

Variables Example

```
1  // operating with variables
2
3  #include <iostream>
4  using namespace std;
5
6  int main ()
7  {
8    // declaring variables:
9    int a
10   float b;
11   int result;
12
13   // process:
14   a = 5;
15   b = 2.3;
16   result = a + b;
17
18   // print out the result:
19   cout << result;
20
21   // terminate the program:
22   return 0;
23 }
```

if else sequence

```
1  if (x == 100)
2  {
3      cout << "x is ";
4      cout << x;
5  }
6  else if (x < 0)
7     cout << "x is negative";
8  else
9     cout << "x is 0";
```

While loop

```
1  // custom countdown using while
2  #include <iostream>
3  using namespace std;
4  int main ()
5  {
6    int n;
7    cout << "Enter the starting number > ";
8    cin >> n;
9    while (n>0) {
10     cout << n << ", ";
11     --n;
12   }
13   cout << "FIRE!\n";
14   return 0;
15 }
```

for loop

```
1
2  for ( n=0, i=100 ; n!=i ; n++, i-- )
3  {
4      // whatever here...
5  }
```

break statement

```
1  // break loop example
2  #include <iostream>
3  using namespace std;
4  int main ()
5  {
6    int n;
7    for (n=10; n>0; n--)
8    {
9      cout << n << ", ";
10     if (n==3)
11     {
12       cout << "countdown aborted!";
13       break;
14     }
15   }
16   return 0;
17 }
```

Output: 10, 9, 8, 7, 6, 5, 4, 3, countdown aborted!

break statement

```
1  // break loop example
2  #include <iostream>
3  using namespace std;
4  int main ()
5  {
6    int n;
7    for (n=10; n>0; n--)
8    {
9      cout << n << ", ";
10     if (n==3)
11     {
12       cout << "countdown aborted!";
13       break;
14     }
15   }
16   return 0;
17 }
```

Output: 10, 9, 8,
7, 6, 5, 4, 3,
countdown aborted!

continue statement

```
1  // continue loop example
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    for (int n=10; n>0; n--) {
8      if (n==5) continue;
9      cout << n;
10   }
11   cout << "FIRE!\n";
12   return 0;
13 }
```

Output: 10, 9, 8, 7, 6, 4, 3, 2, 1, FIRE!

continue statement

```
1  // continue loop example
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    for (int n=10; n>0; n--) {
8      if (n==5) continue;
9      cout << n;
10   }
11   cout << "FIRE!\n";
12   return 0;
13 }
```

Output: 10, 9, 8, 7, 6, 4, 3, 2, 1, FIRE!

Switch-case

```
1   switch (x) {
2     case 1:
3       cout << "x is 1";
4       break;
5     case 2:
6       cout << "x is 2";
7       break;
8     default:
9       cout << "value of x unknown";
10    }
```

```
1   if (x == 1) {
2     cout << "x is 1";
3     }
4   else if (x == 2) {
5     cout << "x is 2";
6     }
7   else {
8     cout << "value of x unknown"
9     }
```

Function and Scope of Variables

```c
#include <stdio.h>
/* global variable declaration */
int a = 20;

int main ()
{
  /* local variable declaration in main function */
  int a = 10;
  int b = 20;
  int c = 0;
  printf ("value of a in main() = %d\n",  a);
  c = sum( a, b);
  printf ("value of c in main() = %d\n",  c);

  return 0;
}
/* function to add two integers */
int sum(int a, int b)
{
     printf ("value of a in sum() = %d\n",  a);
     printf ("value of b in sum() = %d\n",  b);
      {
      int a = 50;
    printf ("value of a in block = %d\n",  a);
      }
     return a + b;
}
```

Output:

value of a in main() = 10
value of a in sum() = 10
value of b in sum() = 20
value of a in block = 50
value of c in main() = 30

Function and Scope of Variables

```
1  #include <stdio.h>
2  /* global variable declaration */
3  int a = 20;
4
5  int main ()
6  {
7    /* local variable declaration in main function */
8    int a = 10;
9    int b = 20;
10   int c = 0;
11   printf ("value of a in main() = %d\n",  a);
12   c = sum( a, b);
13   printf ("value of c in main() = %d\n",  c);
14
15   return 0;
16 }
17 /* function to add two integers */
18 int sum(int a, int b)
19 {
20      printf ("value of a in sum() = %d\n",  a);
21      printf ("value of b in sum() = %d\n",  b);
22       {
23       int a = 50;
24     printf ("value of a in block = %d\n",  a);
25       }
26      return a + b;
27 }
```

Output:

value of a in main() = 10
value of a in sum() = 10
value of b in sum() = 20
value of a in block = 50
value of c in main() = 30

Overloaded Functions

```
1  // overloaded function
2  #include <iostream>
3  using namespace std;
4  int operate (int a, int b)
5  {
6    return (a*b);
7  }
8  float operate (float a, float b)
9  {
10   return (a/b);
11 }
12
13 int main ()
14 {
15   int x=5,y=2;
16   float n=5.0,m=2.0;
17   cout << operate (x,y);
18   cout << "\n";
19   cout << operate (n,m);
20   cout << "\n";
21   return 0;
22 }
```

Output:
10
2.5

Basic Example

```
1  // my first pointer
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int firstvalue, secondvalue;
8    int * mypointer;
9
10   mypointer = &firstvalue;
11   *mypointer = 10;
12   mypointer = &secondvalue;
13   *mypointer = 20;
14   cout << "firstvalue is " << firstvalue << endl;
15   cout << "secondvalue is " << secondvalue << endl;
16   return 0;
17 }
```

Pointers Example 2

```
1  // more pointers
2  #include <iostream>
3  using namespace std;
4  int main ()
5  {
6    int firstvalue = 5, secondvalue = 15;
7    int * p1, * p2;
8    cout << "firstvalue is " << firstvalue << endl;
9    cout << "secondvalue is " << secondvalue << endl;
10   p1 = &firstvalue;  // p1 = address of firstvalue
11   p2 = &secondvalue; // p2 = address of secondvalue
12   *p1 = 10;          // value pointed by p1 = 10
13   *p2 = *p1;         // value pointed by p2 = value pointed by p1
14   cout << "firstvalue is " << firstvalue << endl;
15   cout << "secondvalue is " << secondvalue << endl;
16   p1 = p2;           // p1 = p2 (value of pointer is copied)
17   *p1 = 20;          // value pointed by p1 = 20
18   cout << "firstvalue is " << firstvalue << endl;
19   cout << "secondvalue is " << secondvalue << endl;
20   return 0;
21 }
```

Output: firstvalue is 5
secondvalue is 15
firstvalue is 10
secondvalue is 10
firstvalue is 10
secondvalue is 20

Pointers and Arrays

```
1  // more pointers
2  #include <iostream>
3  using namespace std;
4
5  int main ()
6  {
7    int numbers[5];
8    int * p;
9    p = numbers;  *p = 10;
10   p++;  *p = 20;
11   p = &numbers[2];  *p = 30;
12   p = numbers + 3;  *p = 40;
13   p = numbers;  *(p+4) = 50;
14   for (int n=0; n<5; n++)
15     cout << numbers[n] << ", ";
16   return 0;
17 }
```

Output: 10, 20, 30, 40, 50

Definition

array := &array[0]

```
1  a[5] = 0;       // a [offset of 5] = 0
2  *(a+5) = 0;     // pointed by (a+5) = 0
```

Pointers and Arrays

```
1   // more pointers
2   #include <iostream>
3   using namespace std;
4
5   int main ()
6   {
7     int numbers[5];
8     int * p;
9     p = numbers;  *p = 10;
10    p++;  *p = 20;
11    p = &numbers[2];  *p = 30;
12    p = numbers + 3;  *p = 40;
13    p = numbers;  *(p+4) = 50;
14    for (int n=0; n<5; n++)
15      cout << numbers[n] << ", ";
16    return 0;
17  }
```

Output: 10, 20, 30, 40, 50

Definition

array := &array[0]

```
1   a[5] = 0;       // a [offset of 5] = 0
2   *(a+5) = 0;     // pointed by (a+5) = 0
```

Pointer to functions

```
1  // pointer to functions
2  #include <iostream>
3  using namespace std;
4
5  int addition (int a, int b)
6  { return (a+b); }
7
8  int subtraction (int a, int b)
9  { return (a-b); }
10
11 int operation (int x, int y, int (*functocall)(int,int))
12 {
13   int g;
14   g = (*functocall)(x,y);
15   return (g);
16 }
17
18 int main ()
19 {
20   int m,n;
21   int (*minus)(int,int) = subtraction;
22
23   m = operation (7, 5, addition);
24   n = operation (20, m, minus);
25   cout <<n;
26   return 0;
27 }
```

Returning multiple values from functions

```
1  // more than one returning value
2  #include <iostream>
3  using namespace std;
4
5  void prevnext (int x, int& prev, int& next)
6  {
7    prev = x-1;
8    next = x+1;
9  }
10 int main ()
11 {
12   int x=100, y, z;
13   prevnext (x, y, z);
14   cout << "Previous=" << y << ", Next=" << z;
15   return 0;
16 }
```

Dynamic Memory Allocation

```
1  #include <iostream>
2  using namespace std;
3  int main ()
4  {
5    int i,n;
6    int * p;
7    cout << "How many numbers would you like to type? ";
8    cin >> i;
9    p= new int[i];
10   if (p == 0)
11     cout << "Error: memory could not be allocated";
12   else
13   {
14     for (n=0; n<i; n++)
15     {
16       cout << "Enter number: ";
17       cin >> p[n];
18     }
19     cout << "You have entered: ";
20     for (n=0; n<i; n++)
21       cout << p[n] << ", ";
22     delete[] p;
23   }
24   return 0;
25 }
```