Week# 4
Type Conversion, Bitwise, Logical, and Relational Operators

Dr Taimur Ahmed

Department of IT & CS, PAF-IAST

Lecture# 10
`String` Objects and Stream Manipulators

Working with Characters
`String` Objects

Working with Characters - `cin >>`

- The key word `cin` along with `>>` operator is used to input strings from user, **BUT** it can cause problem too, such as;

It passes over and ignores any leading whitespace characters (spaces, tabs or line breaks)

```
cout << "What is your name?";
cin >> name;  //if Ali Nawaz is entered
cout<< name; // only Ali will be printed
```

Working with Characters – `getline()`

- To **overcome** this limitation, `getline` function can be used to input `string` characters
- **`<string>` header file**

Program Output with Example Input Shown in Bold

```
Please enter your name: Kate Smith [Enter]
Enter the city you live in: Raleigh [Enter]
Hello, Kate Smith
You live in Raleigh
```

```
1  // This program demonstrates using the getline function
2  // to read character data into a string object.
3  #include <iostream>
4  #include <string>
5  using namespace std;
6
7  int main()
8  {
9     string name;
10    string city;
11
12    cout << "Please enter your name: ";
13    getline(cin, name);
14    cout << "Enter the city you live in: ";
15    getline(cin, city);
16
17    cout << "Hello, " << name << endl;
18    cout << "You live in " << city << endl;
19    return 0;
20 }
```

Working with Characters – `getline()`

- To switch between `cin` and `getline()`, use **`cin.ignore()`** function

```
1  //Lecture 10: Working with strings
2  // Example of getline()
3
4  #include<iostream>
5  #include<string>
6  using namespace std;
7
8  int main()
9   {
10     int age;
11     string ch;
12
13     cout << "How old are you: ";
14     cin >> age ;
15
16     cout<<"Enter your name: ";
17
18     cin.ignore();
19
20     getline(cin,ch);
21     cout<< "Hello " << ch << " you are " << age << " years old.";
22
23 return 0;
24  }
```

Program Output:

```
How old are you: 22
Enter your name: Ali Nawaz
Hello Ali Nawaz you are 22 years old.
```

Working with Characters – `cin.get()`

- To read a **single character**:
  - Use `cin`:

```
char ch;
cout << "Strike any key to continue";
cin >> ch;
```

Problem: compiler will skip over **blanks and tabs**

  - Use `cin.get()`:

```
cin.get(ch);
```

Will read the next character entered, **even whitespace**

- It is usually used with array

Working with Characters – `cin.get()`

```
1  // This program demonstrates three ways
2  // to use cin.get() to pause a program.
3  #include <iostream>
4  using namespace std;
5
6  int main()
7  {
8     char ch;
9
10    cout << "This program has paused. Press Enter to continue.";
11    cin.get(ch);
12    cout << "It has paused a second time. Please press Enter again.";
13    ch = cin.get();
14    cout << "It has paused a third time. Please press Enter again.";
15    cin.get();
16    cout << "Thank you!";
17    return 0;
18 }
```

Program Output with Example Input Shown in Bold

```
This program has paused. Press Enter to continue. [Enter]
It has paused a second time. Please press Enter again. [Enter]
It has paused a third time. Please press Enter again. [Enter]
Thank you!
```

Working with Characters – `string`

- To find the **length of a string**:
  - `length()`
  - `size()`

```
1   #include<iostream>
2   #include<string>
3   using namespace std;
4
5   int main()
6   {
7       string text;
8
9       cout << "enter you name ";
10
11      getline(cin,text);
12
13      cout << text.length();
14
15  //  cout << text.size();
16
17      return 0;
18  }
```

Program Output:

```
enter your name  Ali Nawaz

9
```

Note that white space is also counted in character count.

Working with Characters – `string`

- To **concatenate** (join) multiple strings:
  - Use addition (+) operator
  - Combined assignment operator += can also be used

```
string FirstName = "Ali" ;
string LastName = " Nawaz" ;

string FullName = FirstName + LastName;
```

Output is:

```
Ali Nawaz
```

Stream Manipulator

Stream Manipulators

- **Manipulators** are some **operators and functions** which **format output** display of numbers and string etc
- Examples of Stream Manipulators
  - `endl`
  - `fixed`
  - `scientific`

  } Stream Operators

  - `setw()`
  - `setfill()`
  - `setprecision()`

  } Stream Functions
- Requires `iomanip` header file

Stream Manipulators

- `setw()` manipulator
  - This Stream manipulator function sets the **minimum field width** on output
  - Its syntax is `setw(integer)`

```
1   // Lecture 9: Stream Manipulator
2   // setw()
3
4   #include <iostream>
5   using namespace std;
6
7   int main()
8   {
9       cout << 1 << endl << 10 << endl << 100 << endl << 1000;
10
11      return 0;
12  }
```

```
Program Output:
1
10
100
1000
```

```
1   // Lecture 9: Stream Manipulator
2   // setw()
3
4   #include <iostream>
5   #include <iomanip> //header file for setw()
6   using namespace std;
7
8   int main()
9   {
10      cout << setw(5) << 1 << endl <<
11      setw(5) << 10 << endl <<
12      setw(5) << 100 << endl <<
13      setw(5) << 1000;
14
15      return 0;
16  }
```

```
Program Output:
    1
   10
  100
 1000
```

Stream Manipulators

- `setfill()` manipulator
  - This stream manipulator is used **after** `setw()` manipulator function

```
1   // Lecture 9: Stream Manipulator
2   // setfill()
3
4   #include <iostream>
5   #include <iomanip> //header file manipulator
6   using namespace std;
7
8   int main()
9   {
10      cout << setw(5) << setfill('*') << 1 <<endl;
11
12      return 0;
13  }
```

```
Program Output:

****1
```

Stream Manipulators

- `setprecision()` manipulator
  - This stream manipulator is used with **floating point numbers** to set number of digits after decimal
  - It may be used with `fixed` and `scientific`

```
// Lecture 9: Stream Manipulator
// setprecision()

#include <iostream>
#include <iomanip> //header file manipulator
using namespace std;

int main()
{
    float x=0.123456;

    cout << fixed  << x <<endl;
    cout << fixed << setprecision(4)<< x << endl;
    cout << setprecision(3) << x << endl;
    cout << scientific << x << endl;

    return 0;
}
```

Program Output:

```
0.123456
0.1235
0.123
1.235e-001
```

Stream Manipulators

- `setiosflags(iso::format_flag)` Manipulator
  - **Need `<ios>` library header file**

```
#include <ios>
```

- Examples

```
cout<<setiosflags(ios::showpos)<<16;
    +16
cout<<setiosflags(ios::scientific)<<234.564;
    2.34564e+002
cout<<resetiosflags(ios::scientific)<<234.564; // Need to reset after
    234.564
```

Practice Program

- Write a program to generate a grocery purchase receipt, as shown below:

Program Output:

```
Rice------Rs. 100.00
Oil-------Rs. 200.32
Egg-------Rs. 160.00
Bread-----Rs.  80.50
Sugar-----Rs. 400.00
```