Getting Data Into Programs

- An advantage of programs is that they can be run many times using different sets of data.
- A program must have some way of reading the data (this is called input).
- An input stream is an endless sequence of characters coming into your program.
- `cin` is associated with the standard input device (keyboard). It has the extraction operator >>.
- The extraction operator >> takes two operands, `cin` and a variable.

Input Statement Syntax and More

- InputStatement
  cin >> Variable >> Variable ...;
  `cin >> length >> width;` is equal to
  `cin >> length;`
  `cin >> width;`
- Unlike the items specified in an output statement, which can be constants, variables, or complicated expressions, the items specified in an input statement can only be variable names.

Examples Using **cin**

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32

Examples Using **cin**

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32
`cin >> i`	4 60	i=4

Examples Using **cin**

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32
`cin >> i >> j;`	4 60	i=4, j=60

Examples Using **cin**

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32
`cin >> i >> j;`	4 60	i=4, j=60
`cin >> i`	25 A 16.9	i=25

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32
`cin >> i >> j;`	4 60	i=4, j=60
`cin >> i >> ch`	25 A 16.9	i=25, ch='A'

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32
`cin >> i >> j;`	4 60	i=4, j=60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32
`cin >> i >> j;`	4 60	i=4, j=60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i`	25	
	A	
	16.9	i=25

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32
`cin >> i >> j;`	4 60	i=4, j=60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch`	25	
	A	
	16.9	i=25,ch='A'

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i=32
`cin >> i >> j;`	4 60	i=4, j=60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25	
	A	
	16.9	i=25,ch='A',x=16.9

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i = 32
`cin >> i >> j;`	4 60	i = 4, j = 60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25	
	A	
	16.9	i=25,ch='A',x=16.9
`cin >> i`	25A16.9	i=25

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i = 32
`cin >> i >> j;`	4 60	i = 4, j = 60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25	
	A	
	16.9	i=25,ch='A',x=16.9
`cin >> i >> ch`	25A16.9	i=25,ch='A'

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i = 32
`cin >> i >> j;`	4 60	i = 4, j = 60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25	
	A	
	16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25A16.9	i=25,ch='A',x=16.9

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i = 32
`cin >> i >> j;`	4 60	i = 4, j = 60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25	
	A	
	16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25A16.9	i=25,ch='A',x=16.9
`cin >> i`	12 8	i = 12

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i = 32
`cin >> i >> j;`	4 60	i = 4, j = 60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25	
	A	
	16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25A16.9	i=25,ch='A',x=16.9
`cin >> i >> j`	12 8	i = 12, j = 8

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i = 32
`cin >> i >> j;`	4 60	i = 4, j = 60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25	
	A	
	16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25A16.9	i=25,ch='A',x=16.9
`cin >> i >> j >> x;`	12 8	i = 12, j = 8 (Computer waits for a third number)

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i = 32
`cin >> i >> j;`	4 60	i = 4, j = 60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25	
	A	
	16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25A16.9	i=25,ch='A',x=16.9
`cin >> i >> j >> x;`	12 8	i = 12, j = 8 (Computer waits for a third number)
`cin >> i`	46 32.4 15	i = 46

Examples Using `cin`

```
int i, j, k; char ch; float x;
```

Statement	Data	Contents After Input
`cin >> i;`	32	i = 32
`cin >> i >> j;`	4 60	i = 4, j = 60
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25 A 16.9	i=25,ch='A',x=16.9
`cin >> i >> ch >> x;`	25A16.9	i=25,ch='A',x=16.9
`cin >> i >> j >> x;`	12 8	i = 12, j = 8 (Computer waits for a third number)
`cin >> i >> x;`	46 32.4 15	i = 46, x = 32.4 (15 is held for later input)

The Reading Marker and the Newline Character

- The reading marker indicates the next character waiting to be read.
- Each input line has an invisible end-of-line character (the newline character) that tells where one line ends and the next begins.
- A newline character is inserted when you hit an enter or Return or when a program outputs an `endl`. It's referred to as \n.
- Examples:

  `ch = '\n';` **or**

  `cout << "Hello\n";`

  (same as `cout << "Hello" << endl;`**)**

Exploring the Reading Marker

```
int i; char ch; float x;
```

Statements	Contents After Input	Marker Position in the Input Stream
		`25 A 16.9\n`

Exploring the Reading Marker

```
int i; char ch; float x;
```

Statements	Contents After Input	Marker Position in the Input Stream
		`25 A 16.9\n`
`cin >> i;`	i = 25	

Exploring the Reading Marker

```
int i; char ch; float x;
```

Statements	Contents After Input	Marker Position in the Input Stream
		`25 A 16.9\n`
`cin >> i;`	i = 25	
`cin >> ch;`	ch = 'A'	

Exploring the Reading Marker

```
int i; char ch; float x;
```

Statements	Contents After Input	Marker Position in the Input Stream
		`25 A 16.9\n`
`cin >> i;`	i = 25	
`cin >> ch;`	ch = 'A'	
`cin >> x;`	x = 16.9	

Another Reading Marker Example

```
int i; char ch; float x;
```

Statements	Contents After Input	Marker Position in the Input Stream
		25\n A\n 16.9\n

Another Reading Marker Example

```
int i; char ch; float x;
```

Statements	Contents After Input	Marker Position in the Input Stream
		25\n A\n 16.9\n
`cin >> i;`	i = 25	

Another Reading Marker Example

```
int i; char ch; float x;
```

Statements	Contents After Input	Marker Position in the Input Stream
		25\n A\n 16.9\n
`cin >> i;`	i = 25	
`cin >> ch;`	ch = 'A'	

Another Reading Marker Example

```
int i; char ch; float x;
```

Statements	Contents After Input	Marker Position in the Input Stream
		25\n A\n 16.9\n
`cin >> i;`	i = 25	
`cin >> ch;`	ch = 'A'	
`cin >> x;`	x = 16.9	

Reading Characters withthe **get** Function

- The >> operator skips any leading white-space characters (such as blanks and new-line characters) while looking for the next data value in the input stream.
- Consider **`cin >> ch1 >> ch2;`** and an input stream of **R 1.** ch1 = 'R' ch2 = '1' What if you wanted the blank also? You can't do it with the extraction operator.
- You can use the get function, which doesn't skip any characters. **`cin.get(someChar);`**
- The argument to the get function must be a character variable.

The **get** Function versus the Extraction Operator

```
char ch1, ch2, ch3;
```

Statements	Contents After Input	Marker Position in the Input Stream
		A B\n CD\n
`cin >> ch1;`	ch1 = 'A'	
		A B\n CD\n
`cin >> ch1;`	ch1 = 'A'	

The **get** Function versus the Extraction Operator

```
char ch1, ch2, ch3;
```

Statements	Contents After Input	Marker Position in the Input Stream
		A B\n CD\n
cin >> ch1; cin >> ch2;	ch1 = 'A' ch2 = 'B'	
		A B\n CD\n
cin >> ch1; cin >> ch2;	ch1 = 'A' ch2 = 'B'	

The **get** Function versus the Extraction Operator

```
char ch1, ch2, ch3;
```

Statements	Contents After Input	Marker Position in the Input Stream
		A B\n CD\n
cin >> ch1; cin >> ch2; cin >> ch3;	ch1 = 'A' ch2 = 'B' ch3 = 'C'	
		A B\n CD\n
cin >> ch1; cin >> ch2; cin.get(ch3);	ch1 = 'A' ch2 = 'B' ch3 = '\n'	

Skipping Characters with the **ignore** Function

- The **ignore** function is used to skip charac-ters in the input stream.
- It's a function with two arguments
  `cin.ignore(200, '\n');`
- The first argument is **int**, the second **char**. This statement says skip 200 characters or skip until you reach a newline character.

Reading String Data

- The extraction operator can be used but not with any strings which have blanks embedded in them. (it stops reading when it encounters a whitespace)
- There's a function we can use: **getline.**
- It stops when it reaches a newline character. (the newline character is consumed)

  Example: `getline(cin, myString);`

Interactive Input/Output

```
//*****************************************************************
// Prompts program
// This program demonstrates the use of input prompts
//*****************************************************************
#include <iostream>
#include <iomanip>    // For setprecision()

using namespace std;

int main()
{
    int   partNumber;
    int   quantity;
    float unitPrice;
    float totalPrice;

    cout << fixed << showpoint            // Set up floating-pt.
         << setprecision(2);              //   output format
```


Interactive Input/Output

```
  cout << "Enter the part number:" << endl;          // Prompt
  cin >> partNumber;

  cout << "Enter the quantity of this part ordered:" // Prompt
       << endl;
  cin >> quantity;

  cout << "Enter the unit price for this part:"      // Prompt
       << endl;
  cin >> unitPrice;

  totalPrice = quantity * unitPrice;
  cout << "Part " << partNumber                      // Echo print
       << ", quantity " << quantity
       << ", at $ " << unitPrice << " each" << endl;
  cout << "totals $ " << totalPrice << endl;
  return 0;
}
```

Interactive I/O Example

```
Enter the part number:
4671
Enter the quantity of this part ordered:
10
Enter the unit price for this part:
27.25
Part 4671, quantity 10, at $ 27.25 each
Totals $ 272.50
```