We are 183

L03: Week 2 - Wednesday

Computing Carnival!

Last Time… on EECS 183

Basic I/O
Comments
Data Types
Basic Arithmetic
Operators
Source Code

Data Types

- bool true, false
- int 5, 20, -42
- double 5.42, 5.4E3, -56.0
- char 'A', '9'
- string "This is cool!"

Operators

- + -
- * / %

iClicker

- Does the expression correctly capture the intent?

- Intent: itemsA + itemsB, divided by 2

- Expression:
- itemsA + itemsB / 2

A)Yes
B)No

iClicker

- Does the expression correctly capture the intent?

- Intent: itemsA + itemsB, divided by 2

- Expression:
- itemsA + itemsB / 2
- (itemsA + itemsB) / 2

A)Yes
B)No

Max size of an int

- Little over 2 billion
- 2,xxx,xxx,xxx

- That's 10 digits
- 1st one is a 2

You really don't want to spill that…

You can't put a large value into a small cup

- Well, OK, you can, but you'll lose some
- called "spillage" or "overflow"
- Compiler tries to help prevent this
  - issues warning
  - pay attention to all "warnings"

iClicker

Will

```
int x = 1234567890;
```

cause an overflow?

A)Yes

B)No

iClicker

Will

```
int x = 1234567890;
```

cause an overflow?

A)Yes

B)No

iClicker

Will

```
int x = 4000000000;
```

cause an overflow?

A)Yes

B)No

iClicker

Will

```
int x = 4000000000;
```

cause an overflow?

A)Yes

B)No

iClicker

Will

```
int x = 1000;
int y = 2000;
int z = x * y * x * 2;
```

cause an overflow?

A)Yes

B)No

iClicker

Will

```
int x = 1000;
int y = 2000;
int z = x * y * x * 2;
```

cause an overflow?

A)Yes

B)No

iClicker

Will

```
int x = 1000;
int y = 2000;
double z = x * y * x * 2;
```

cause an overflow?

A)Yes

B)No

iClicker

Will

```
int x = 1000;
int y = 2000;
double z = x * y * x * 2;
```

cause an overflow?

A)Yes

B)No

Operator Precedence Rules

OPERATOR	ASSOCIATIVITY
( )	left to right
+x -x pos/neg	unary; right to left
* / %	left to right
+ - (add, subtract)	left to right
=	right to left

HIGH

LOW

iClicker (midterm1 – winter 12)

If an integer variable x has been initialized to 7, which of the following code fragments will assign the number 8 to x?

```
A) x = x + 1;
B) x - 1 = 7;
C) x << 8;
D) x = '8';
```

E) all of the above

iClicker (midterm1 – winter 12)

If an integer variable x has been initialized to 7, which of the following code fragments will assign the number 8 to x?

A) `x = x + 1;`
B) `x - 1 = 7;`
C) `x << 8;`
D) `x = '8';`
E) all of the above

iClicker (midterm1 – Fall 08)

Consider the following code fragment:

```
double x = 17.0 / 2 - 5 / 2;
cout << x;
```

What does the above code print?

A) -2
B) 2
C) 6
D) 6.5
E) the above code causes a division by zero error

iClicker (midterm1 – Fall 08)

Consider the following code fragment:

8.5 – 2

```
double x = 17.0 / 2 - 5 / 2;
cout << x;
```

What does the above code print?

A) -2
B) 2
C) 6
D) 6.5
E) the above code causes a division by zero error

Today

Casting, Imprecision
Compile and run-time Errors
Testing & Debugging
Pre-defined Functions
more on cin, cout

Fun with Casting

```
double x = 3.7;

int i = (int) (x);

i = (int) (x + 0.5);
```

Variables and Types

```
#include <iostream>
using namespace std;

int main() {
    int pies = 0;
    int people = 0;
    cout << "Number of pies: ";
    cin >> pies;
    cout << "Number of people: ";
    cin >> people;
    cout << 360 * (pies / people);
    cout << " degrees of pie per person!";
    cout << endl;
}
```

Variables and Types

```
int main() {
    int pies = 0;
    int people = 0;
    cout << "Number of pies: ";
    cin >> pies;
    cout << "Number of people: ";
    cin >> people;
    cout << 360 * (pies / people);
    cout << " degrees of pie per person!";
    cout << endl;
}
```

Variables store values of a specific type. This variable stores integers, and is first set to 0.

Variables and Types

```
int main() {
    int pies = 0;
    int people = 0;
    cout << "Number of pies: ";
    cin >> pies;
    cout << "Number of people: ";
    cin >> people;
    cout << 360 * (pies / people);
    cout << " degrees of pie per person!";
    cout << endl;
}
```

This replaces the value of pies with whatever the user types.

Variables and Types

```
int main() {
    int pies = 0;
    int people = 0;
    cout << "Number of pies: ";
    cin >> pies;
    cout << "Number of people: ";
    cin >> people;
    cout << 360 * (pies / people);
    cout << " degrees of pie per person!";
    cout << endl;
}
```

Output:

```
Number of pies: 3
Number of people: 2
360 degrees of pie per
person!
```

Variables and Types

```
int main() {
    int pies = 0;
    int people = 0;
    cout << "Number of pies: ";
    cin >> pies;
    cout << "Number of people: ";
    cin >> people;
    cout << 360 * (pies / people);
    cout << " degrees of pie per person!";
    cout << endl;
}
```

Output:

```
Number of pies: 3
Number of people: 2
360 degrees of pie per
person!
```

Variables and Types

```
int main() {
    double pies = 0;
    int people = 0;
    cout << "Number of pies: ";
    cin >> pies;
    cout << "Number of people: ";
    cin >> people;
    cout << 360 * (pies / people);
    cout << " degrees of pie per person!";
    cout << endl;
}
```

Solution 1: make pies a double

```
Number of pies: 3
Number of people: 2
540 degrees of pie per person!
```

Variables and Types

```
int main() {
    int pies = 0;
    double people = 0;
    cout << "Number of pies: ";
    cin >> pies;
    cout << "Number of people: ";
    cin >> people;
    cout << 360 * (pies / people);
    cout << " degrees of pie per person!";
    cout << endl;
}
```

Solution 2: make people a double

```
Number of pies: 3
Number of people: 2
540 degrees of pie per person!
```

Variables and Types

```
int main() {
    int pies = 0;
    int people = 0;
    cout << "Number of pies: ";
    cin >> pies;
    cout << "Number of people: ";
    cin >> people;
    cout << 360 * (double) pies / people;
    cout << " degrees of pie per person!";
    cout << endl;
}
```

Solution 3: cast pies/people to a double

```
Number of pies: 3
Number of people: 2
540 degrees of pie per person!
```

Variables and Types

```
double result = 360 * (double) pies / people;

cout << result
     << " degrees of pie per person!";
    cout << endl;
}
```

Solution 4: assign into a double variable

```
Number of pies: 3
Number of people: 2
540 degrees of pie per person!
```

Variables and Types

```
double result = 360 * (double) pies / people;

cout << result
     << " degrees of pie per person!";
    cout << endl;
}
```

Solution 4: cast result back to an int

```
Number of pies: 3
Number of people: 7
154.28 degrees of pie per person!
```

Difficult to cut

Variables and Types

```
int result = (int)(360.0 * pies / people);

cout << result
     << " degrees of pie per person!";
    cout << endl;
}
```

Solution 4: cast result back to an int

```
Number of pies: 3
Number of people: 7
154 degrees of pie per person!
```

From Zyante:

- A **function** is a list of statements that can be executed by referring to the function's name
- An input value to the function appears between ( )
- The function executes and returns a new value

Example: Pythagorean Theorem

- x = square-root-of($y^2 + z^2$)

```
double x = sqrt(y * y + z * z);
```

Example: Pythagorean Theorem

- x = square-root-of($y^2 + z^2$)

- A **function** is a list of statements that can be executed by referring to the function's name

```
double x = sqrt(y * y + z * z);
```

Example: Pythagorean Theorem

- x = square-root-of($y^2 + z^2$)

- **An input value to the function appears between ( )**
- int y = 3;
- int z = 4;

```
double x = sqrt(y * y + z * z); // 3 * 3 + 4 * 4
double x = sqrt(25);
```

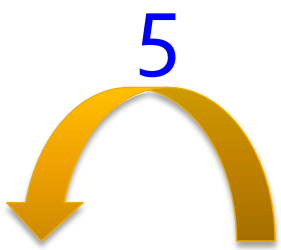
Example: Pythagorean Theorem

- x = square-root-of($y^2 + z^2$)
- The function **<u>executes</u>** and **<u>returns a new value</u>**

```
double x = sqrt(25);
```

Example: Pythagorean Theorem

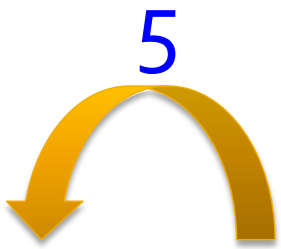
- x = square-root-of($y^2 + z^2$)
- The function **<u>executes</u>** and **<u>returns a new value</u>**




```
double x = sqrt(25);
```

#include <cmath>

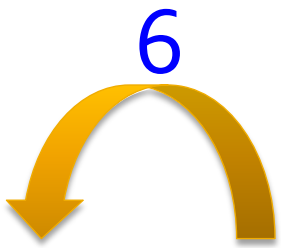
- A **<u>function</u>** is a list of statements that can be executed by referring to the function's name
- An input value to the function appears between ( )
- The function executes and returns a new value



double x = **abs(-5);**

#include <cmath>

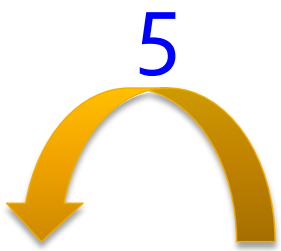
- A **<u>function</u>** is a list of statements that can be executed by referring to the function's name
- An input value to the function appears between ( )
- The function executes and returns a new value



double x = **ceil(5.2);**

#include <cmath>

- A **<u>function</u>** is a list of statements that can be executed by referring to the function's name
- An input value to the function appears between ( )
- The function executes and returns a new value



double x = **floor(5.2);**

<cmath> Functions

- Assume number of students in lecture is 170
- Each students gets a cupcake
- 1 pack holds 6 cupcakes

- How many packs are needed?

cout

cout << Examples

```
cout << "One";
```

Console

```
One
```

cout << Examples

```
cout << "One" << "Two";
```

Console

```
OneTwo
```

cout << Examples

```
cout << "One" << "  Two";
```

Notice the space

Console

```
One Two
```

cout << Examples

```
cout << "One"
     << "  Two";
```

Console

```
One Two
```

cout << Examples

```
cout << "One"
     << "  Two"
     << "  Three";
```

Console

```
One Two Three
```

cout << Examples

```
cout << "One"
     << " Two"
     <<

" Three";
```

Console

```
One Two Three
```

cout << Examples

```
cout << "Hello World!";
```

Console

```
Hello World!
```

cout << Examples

```
cout << "Hello World!";
cout << "C++ is amazing";
```

Console

```
Hello World!C++ is amazing
```

cout << Examples

```
cout << "Hello World!"<< endl;
cout << "C++ is amazing";
```

Console

```
Hello World!
C++ is amazing
```

cout << Examples

```
cout << "Hello there";
cout << endl << endl;
cout << "Bye";
```

Console

```
Hello there

Bye
```

cin

The extraction operator >>

- Skips over “ignores” leading whitespace characters
  - Spaces
  - Tabs
  - new lines

  - Actually it reads but does not store anywhere

The extraction operator >>

- Ignores leading white spaces
- Reads one `char` at a time
- Stops reading when it hits a `char` that is NOT acceptable to the datatype it's looking for
- Converts chars read to the expected datatype

The extraction operator >>

```
int age;

cin >> age;
```

Console

```
_
```

The extraction operator >>

```
int age;

cin >> age;
```

Console

```
25 <enter>
```

The extraction operator >>

```
int age;

cin >> age;
```

Same results

Console

```
25 <enter>
```

The extraction operator >>

```
int age;

cin >> age;
```

Same results

age

Console

```
<tab>25 <enter>
```

The extraction operator >>

```
int age;

cin >> age;
```

Same results

age

Console

25A<enter>

Expecting an int,
so stops reading
at the first non-int

The extraction operator >>

```
int age;

cin >> age;
```

Same results

age

Console

25 35<enter>

Stops at the whitespace

The extraction operator >>

```
int age;

cin >> age;
```

Same results

age

Console

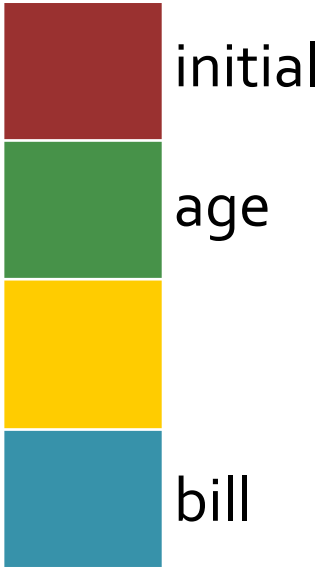
25 35<enter>

25 will be stored in "age" and 35 will stay in the input buffer

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

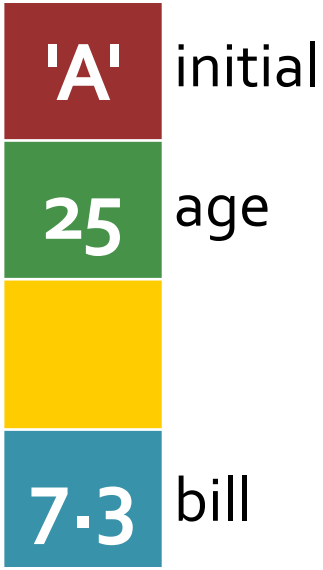


Console

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```



Console

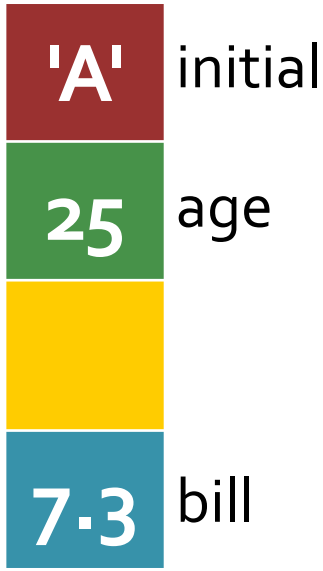
```
25<enter>
A<enter>
7.3<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Same results



Console

```
25  A  7.3<enter>
```

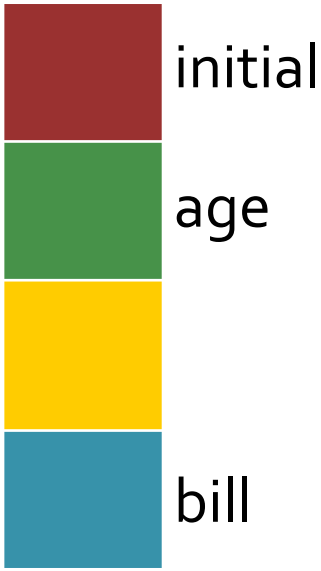
The extraction operator >>

Execution

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Let's how reading works



Console

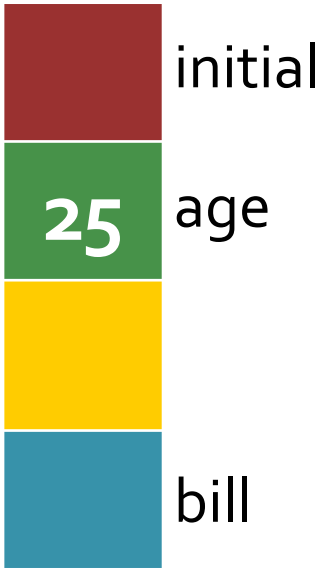
```
25  A  7.3<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

25 A 7.3<enter>

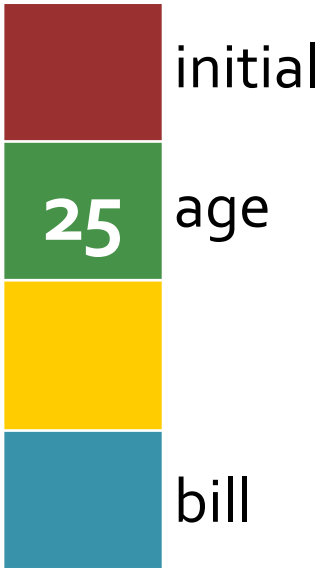
The reading marker

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

25 A 7.3<enter>

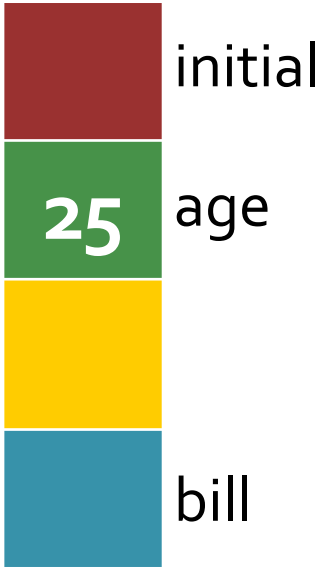
Stops at space and reads 25

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

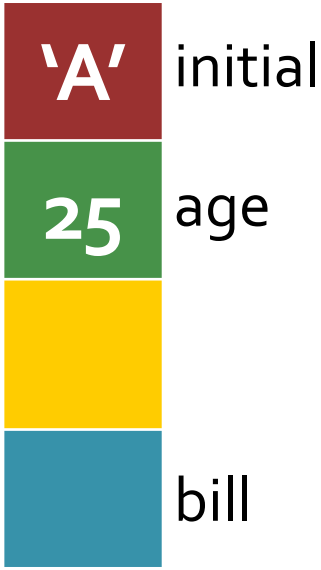
25 A 7.3<enter>

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

25 A 7.3<enter>

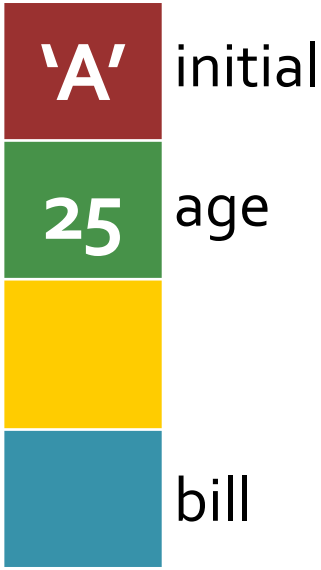
Stops at space and reads 'A'

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

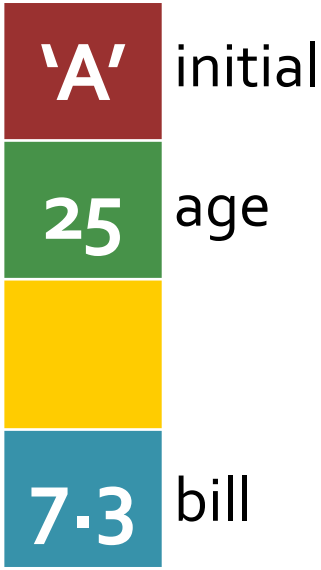
25 A 7.3<enter>

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

25 A 7.3<enter>

Stops at <enter> and reads 7.3

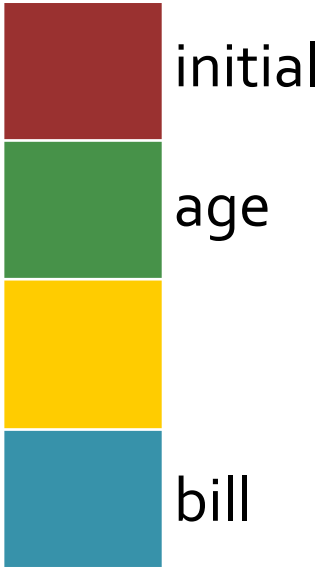
The extraction operator >>

Execution

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Let's consider a different input



Console

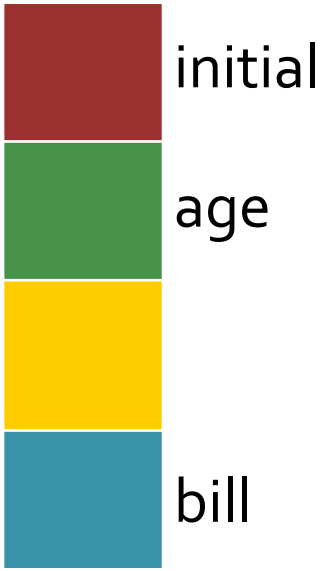
```
29  16.9A<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

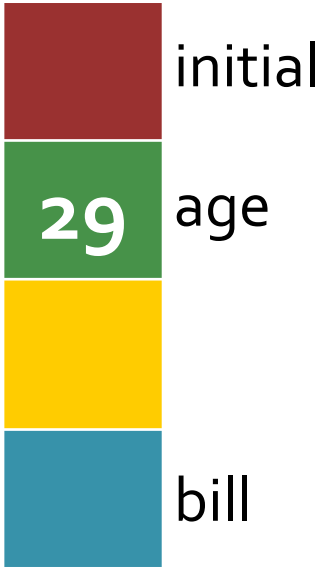
```
29  16.9A<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

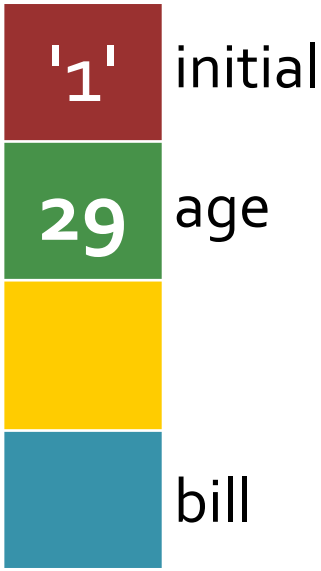
```
29 16.9A<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

29 16.9A<enter>

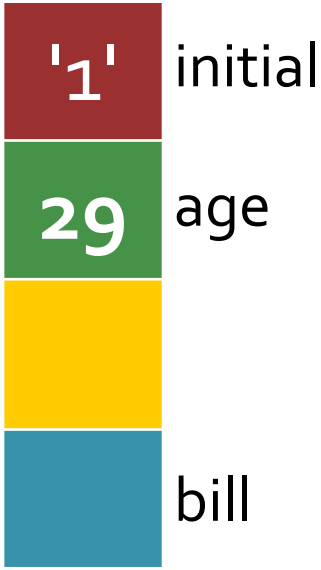
Expecting a char, so reads exactly one char

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

```
29  16.9A<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution

'1'	initial
29	age
6.9	bill

Console

29 16.9A<enter>

Stops at 'A' since it's a different datatype

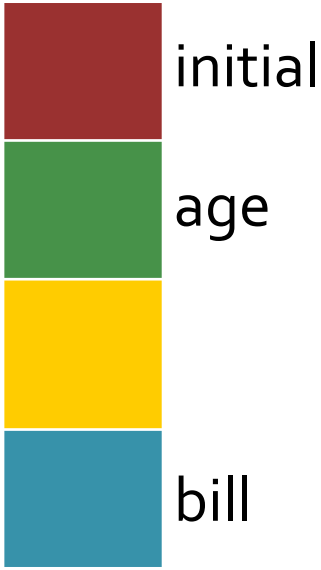
The extraction operator >>

Execution

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Let's consider a different input



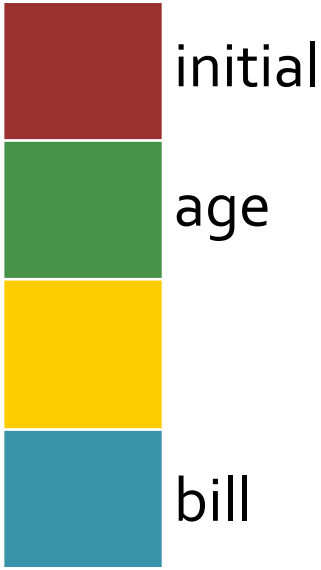
Console

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

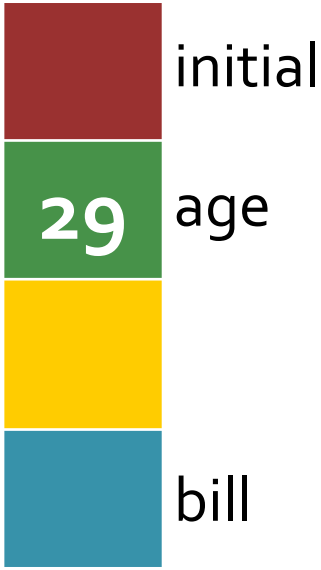
```
29  1A<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

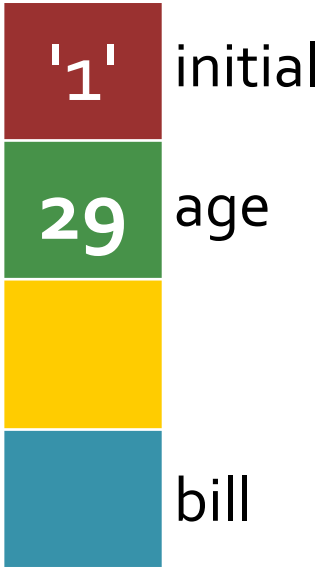
```
29 1A<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Execution



Console

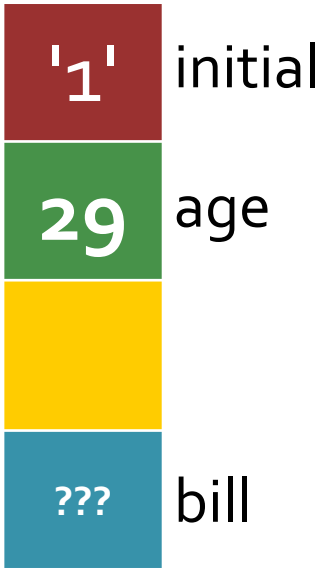
```
29  1A<enter>
```

The extraction operator >>

```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Why?



Execution

Console

29 1A<enter>

Reads double.
Stores an arbitrary
(garbage) value in bill

The extraction operator >>

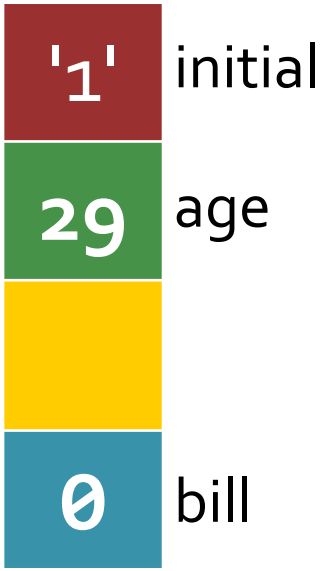
```
int age;
char initial;
double bill;

cin >> age;
cin >> initial;
cin >> bill;
```

Why?

double size is 8 bytes

'A' is 1 byte



Execution

Console

29 1A<enter>

Reads double.
Stores a value of 0 in bill

cin enters *fail state*

- When
  - datatype of value read doesn't match datatype of variable

<u>CRITICAL to understand</u>

- NO reading takes place again until cleared

Reading Strings

- Using a string object

```
#include <string>
#include <iostring>
using namespace std;

int main()
{
    string    message;
    cin   >>  message;
    cout  <<  message;
    return 0;
}
```

We need to include the string the library

Console

```
Hello<enter>
Hello
```

Reading Strings

- Using a string object

```
#include <string>
#include <iostring>
using namespace std;

int main()
{
    string    message;
    cin   >>  message;
    cout  <<  message;
    return 0;
}
```

Execution

Console

```
Hello World<enter>
```

Reading Strings

- Using a string object

```
#include <string>
#include <iostring>
using namespace std;

int main()
{
    string    message;
    cin   >>  message;
    cout  <<  message;
    return 0;
}
```

Execution

Console

Hello World<enter>
Hello

Why not "Hello World"?

Reading Strings

Because: As always:

- >> operator skips any leading whitespace
- reads successive char's into the string
- stops at the first trailing whitespace
- which is not consumed, but remains waiting in the input stream

Reading Strings

What if we want to read a string that includes a whitespace
(e.g. "Hello World") ?

Answer: use getline()

getline()

- **Usage:** getline (<stream>, <stringvar>);
  - A function that takes two parameters
- leading whitespace NOT skipped
- reading stops when newline(\n) encountered
- Then read pointer proceeds to after <\n>
- <\n> is not put into the string

getline() example

Execution

```
string message;
getline(cin, message);
```

Console

Hello World!\n

\n = <enter>

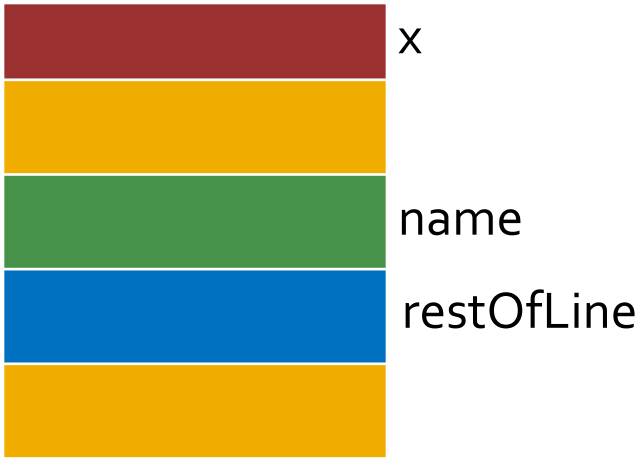
cin>> combo with getline()

Execution

```
int x;
string restOfLine, name;
cout << "Enter data: ";

cin >> x;
getline(cin,restOfLine);
getline(cin, name);

cout << "x is: " << x << endl << "name: " << name;
```



Console

Enter Data:

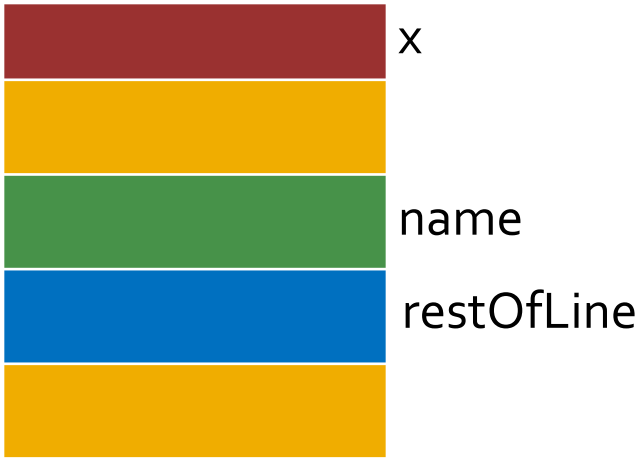
cin>> combo with getline()

Execution

```
int x;
string restOfLine, name;
cout << "Enter data: ";

cin >> x;
getline(cin,restOfLine);
getline(cin, name);

cout << "x is: " << x << endl << "name: " << name;
```



Console

```
Enter Data: 4 don't want this\n
Indiana Jones\n
```

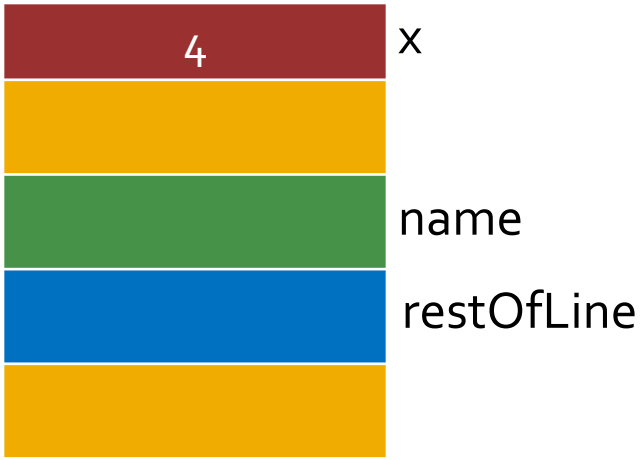
cin>> combo with getline()

```
int x;
string restOfLine, name;
cout << "Enter data: ";

cin >> x;
getline(cin,restOfLine);
getline(cin, name);

cout << "x is: " << x << endl << "name: " << name;
```

Execution



Console

```
Enter Data: 4 don't want this\n
Indiana Jones\n
```

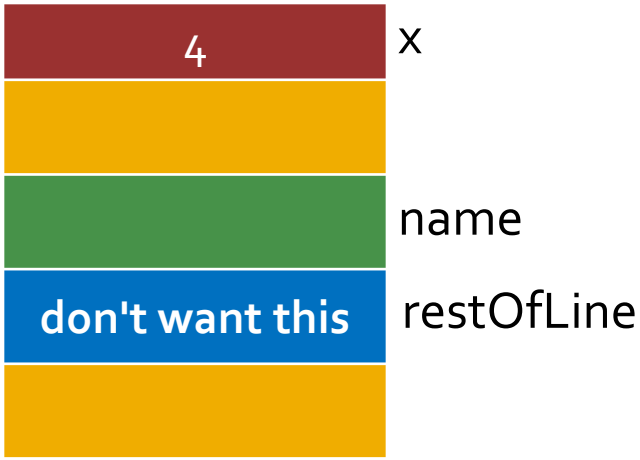
cin>> combo with getline()

```
int x;
string restOfLine, name;
cout << "Enter data: ";

cin >> x;
getline(cin,restOfLine);
getline(cin, name);

cout << "x is: " << x << endl << "name: " << name;
```

Execution



Console

```
Enter Data: 4 don't want this\n
Indiana Jones\n
```

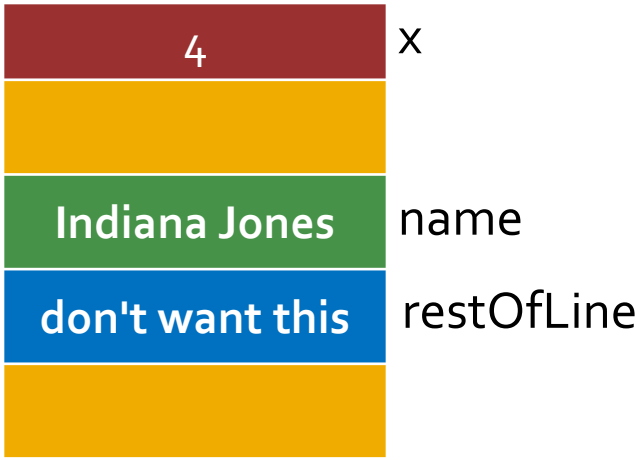
cin>> combo with getline()

```
int x;
string restOfLine, name;
cout << "Enter data: ";

cin >> x;
getline(cin,restOfLine);
getline(cin, name);

cout << "x is: " << x << endl << "name: " << name;
```

Execution



Console

Enter Data: 4 don't want this\n

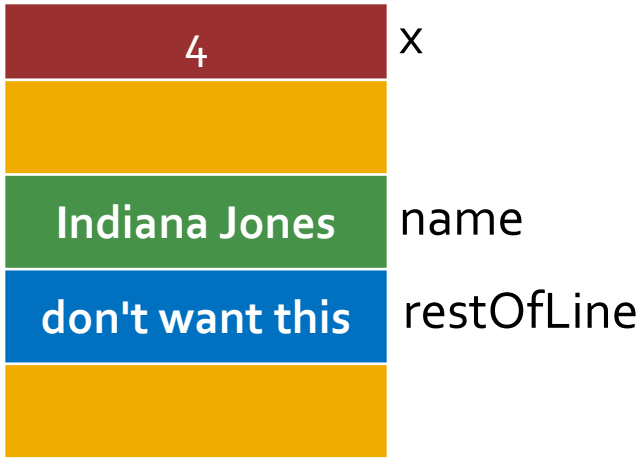
Indiana Jones\n

cin>> combo with getline()

```
int x;
string restOfLine, name;
cout << "Enter data: ";

cin >> x;
getline(cin,restOfLine);
getline(cin, name);

cout << "x is: " << x << endl << "name: " << name;
```



Execution

Console

```
Enter Data: 4 don't want this\n
Indiana Jones\n
x is: 4
name: Indiana Jones
```

ignore()

- Skips (reads and discards) char's in the input stream
- This function is RARELY USED
- used if data has "columns"
- otherwise, DO NOT USE

ignore()

- Skips (reads and discards) char's in the input stream
- This function is RARELY USED
- NEVER for this class

ignore()

- Usage:

```
cin.ignore(howMany, whatChar);
```

- will skip over the **howMany** char's or until **whatChar** has been read whichever comes first

Project #1

Cupcakes

Project #1

- Due in little over one week:
  Friday, January 22nd

Project #1

- Due in little over one week:
  Friday, January 22nd

- Extra Credit:
  - +5% if your last submit is 2 days before due date
  - +2.5% if your last submit is the day before

- Another great reason to **start early**

Autograder

- Gives "full disclosure" on 1st test case missed
- This is your final score on runnability
- Infinite number of submits
- Feedback on 1st two per day
- Once per project, feedback on a 3rd submit

A word on Honesty

Honor Code Violations

- Submitted to Honor Council
  - Council of your peers
- You **CANNOT** drop the course
- Just one common penalty (for one violation):
  - 0 on the project
  - -1 letter grade on your final class grade
  - **and** Community service

That's All Folks!

Remember:

Discussion Sections
Assignment 1 Due **this Friday (codelab, survey)**
Start Project 1 **early**
Office Hours for help when you're stuck

Review Questions

Question 1 (midterm1 – winter 12)

If an integer variable x has been initialized to 7, which of the following code fragments will assign the number 8 to x?

A) x = x + 1;
B) x - 1 = 7;
C) x << 8;
D) x = '8';
E) all of the above

Question 1 (midterm1 – winter 12)

If an integer variable x has been initialized to 7, which of the following code fragments will assign the number 8 to x?

A) x = x + 1;
B) x - 1 = 7;
C) x << 8;
D) x = '8';
E) all of the above

Question 2 (midterm1 – Fall 08)

Consider the following code fragment:

```
double x = 17.0 / 2 – 5 / 2;
cout << x;
```

What does the above code print?

a) -2
b) 2
c) 6
d) 6.5
e) the above code causes a division by zero error

Question 2 (midterm1 – Fall 08)

Consider the following code fragment:

8.5 – 2

```
double x = 17.0 / 2 – 5 / 2;
cout << x;
```

What does the above code print?

a) -2
b) 2
c) 6
d) 6.5
e) the above code causes a division by zero error

Question 3 (midterm1 – fall 12)

Consider the following code fragment:

```
double x=2, y=4;
char ch;
cin >> x >> ch >> y;
cout << x * y;
```

A. 4
B. 5
C. 6
D. 8
E. 10

what does it print given the user input shown below?

Console

```
2.5*2.0<Enter>
```

Question 3 (midterm1 – fall 12)

Consider the following code fragment:

```
double x=2, y=4;
char ch;
cin >> x >> ch >> y;
cout << x * y;
```

A. 4
B. 5
C. 6
D. 8
E. 10

what does it print given the user input shown below?

Console

```
2.5*2.0<Enter>
```

Question 4 (midterm1 – fall 12)

Consider the following code fragment:

```
int x=2, y=4;
char ch;
cin >> x >> ch >> y;
cout << x * y;
```

A. 4
B. 5
C. 6
D. 8
E. 10

what does it print given the user input shown below?

Console

```
2.5*2.0<Enter>
```

Question 4 (midterm1 – fall 12)

Consider the following code fragment:

```
int x=2, y=4;
char ch;
cin >> x >> ch >> y;
cout << x * y;
```

A. 4
B. 5
C. 6
D. 8
E. 10

what does it print given the user input shown below?

Console

```
2.5*2.0<Enter>
```

Question 5 (midterm1 – winter 12)

Consider the following code fragment:

```
int  i1, i2, i3, i4;
char c1, c2, c3;
cin  >> i1 >> c1 >> i2 >> c2 >> i3 >> c3 >> i4;
cout << i1 << c1 << i2 << c2 << i3 << c3 << i4;
```

a) 1+2.3+4+5.6
b) 1 + 2.3 + 4 + 5.6
c) 1+2+4+5
d) 1 + 2 + 4 + 5
e) 1+2.3+4

what does it print given the user input shown below?

Console

```
1+2.3+4+5.6<Enter>
```

Question 5 (midterm1 – winter 12)

Consider the following code fragment:

```
int  i1, i2, i3, i4;
char c1, c2, c3;
cin  >> i1 >> c1 >> i2 >> c2 >> i3 >> c3 >> i4;
cout << i1 << c1 << i2 << c2 << i3 << c3 << i4;
```

a) 1+2.3+4+5.6
b) 1 + 2.3 + 4 + 5.6
c) 1+2+4+5
d) 1 + 2 + 4 + 5
e) 1+2.3+4

what does it print given the user input shown below?

Console

```
1+2.3+4+5.6<Enter>
```

Question 6 (midterm1 – winter 12)

Consider the following code fragment:

```
cout << "\\\";  // Is this a comment?";
```

What does the above code fragment print?

a) \\\
b) \"; // Is this a comment?
c) \\"; // Is this a comment?
d) \\\"; // Is this a comment?
e) "\\\"; // Is this a comment?"

Question 6 (midterm1 – winter 12)

Consider the following code fragment:

```
                \   "
cout << "\\\";  // Is this a comment?";
```

\\ prints \
\" prints " as a char

What does the above code fragment print?

a) \\\
b) \"; // Is this a comment?
c) \\"; // Is this a comment?
d) \\\"; // Is this a comment?
e) "\\\"; // Is this a comment?"

Common Assignment Errors

```
double x(5.2), y(7.5);
char ch = 'A';
int i = 10, j = 42;

x = x % y;

ch = $;

i = 5;
j = 0;
x = i / j;
```

Common Assignment Errors

```
double x(5.2), y(7.5);
char ch = 'A';
int i = 10, j = 42;

x = x % y;          // syntax error

ch = $;             // syntax error

i = 5;
j = 0;
x = i / j;          // run-time error
```

Sharpen your pencil

- From the following list, circle the statements that would be legal if these lines were in a single main program

```
int x = 34.5;
bool boo = x;
int g = 17;
int y = g;
y = y + 10;
int s;
s = y;
int b = 3;
int v = b;
int n = 12;
v = n;
double k = 1000 * 1000 * 1000 * 90 * .5;
double y = 9.5;
int p = 3 * g + y;
```

Basics

Across

3. Can't pin it down
4. Acronym for a chip
7. What's a prompt good for?
8. Just gotta have one
10. RUN
13. You're never going to change!!!
14. Could be called "Father"

Down

1. Quite a crew of characters
2. Not an integer (or _____ your boat)
3. Nothing is there
5. Source code consumer
6. Acronym for your laptop's power
9. Announce a new variable
11. Number variable type
12. Department of LAN jockeys
13. Say something