COMSC-110 Chapter 5

Console IO

Valerie Colber, MBA, PMP, SCPM

Sources of Input

- Set fixed values in a program
- From the keyboard (interactive)
- From a file (interactive)

Console Output

1. Program introduction
2. Label = text output that says what the following value output is
3. Values
4. Prompt = text output that says what data to input

Interactive programs

- Interactive programs are designed for a certain amount of interaction to take place between the user and the program.
- This interaction transfers data values into variables, *after* the program has been written and compiled!
- Two sources of interactive input are the keyboard and text files.

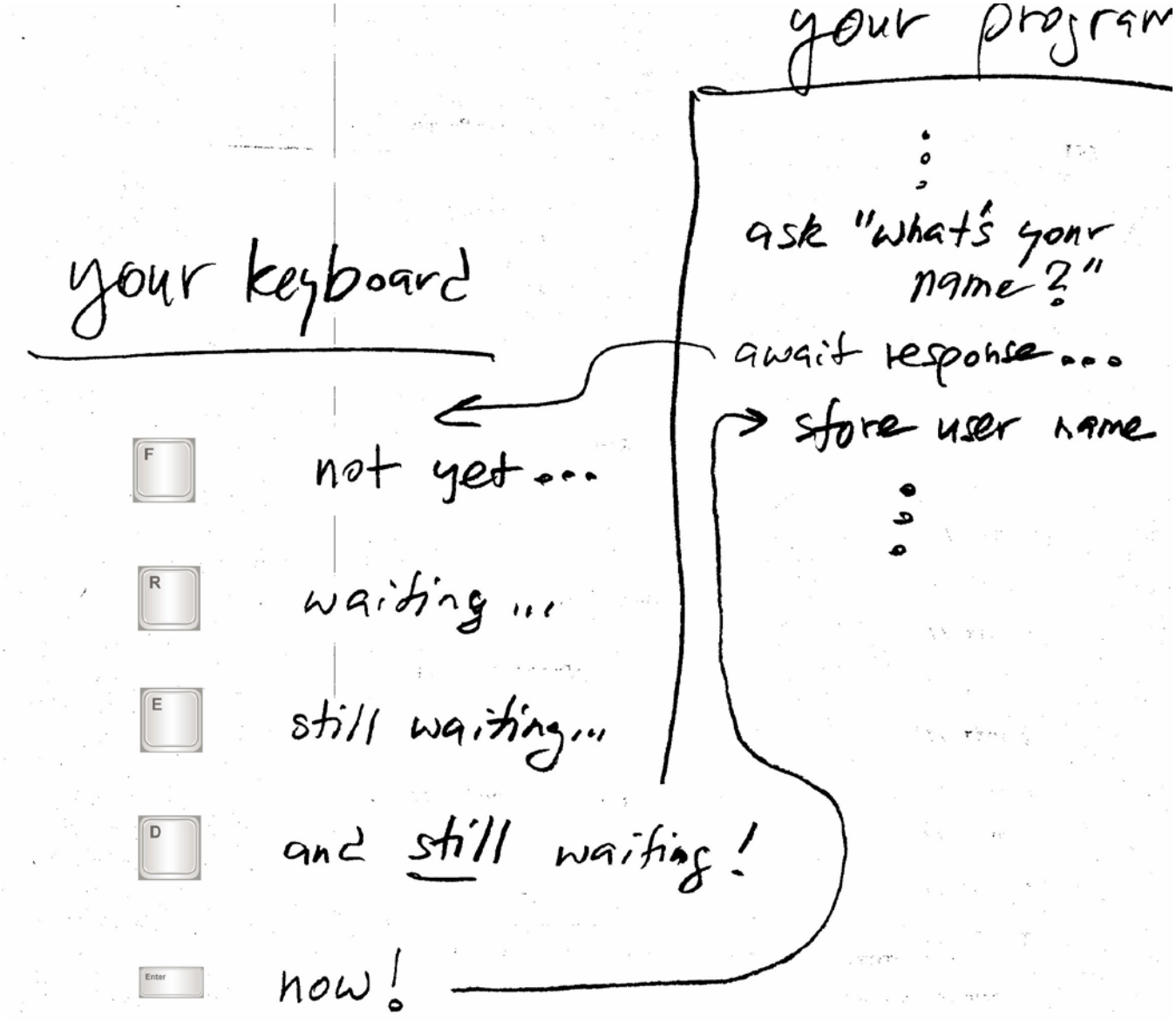
The Input/Output (IO) Libraries

- #include <iostream> // for console IO
- #include <fstream> // for file IO
  using namespace std;

Capturing values from the keyboard

- Reserve a place in the computer's memory to store the input value.
- Prompt the user to type an input value and press the ENTER key.
- Transfer the keyboard entry into the memory reserved for this use.

Capturing values from the keyboard



Capturing values from the keyboard

keyboard "buffer" grows ...

keyboard buffer contents sent to the program :

when the ENTER key is pressed

program "reads" the contents

Capturing values from the keyboard

programmer input

```
int age = 21;
double amountBorrowed = 500000;
```

Console input (2 Examples: integer and decimal)

1. int age;
2. prompt user to enter his age
3. transfer the typed value into the variable "age"

1. double amountBorrowed;
2. prompt user to enter mortgage balance
3. transfer the typed value into the variable "amountBorrowed"

Using programmer input

```
…
//libraries
#include <iostream>
using namespace std;
…
//main program
int main()
{
  //data
  int a = 3;   //first number
  int b = 4;  //second number
  int c = a + b;  //sum of first number and second number

  //output results
  cout << "Simple addition with programmer input" << endl;
  cout << a << " plus " << b << " equals " << c << endl;
}//main
```

Using console input

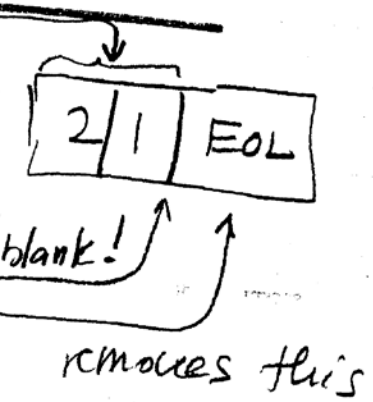
add2Numbers.cpp

Extract and discard characters

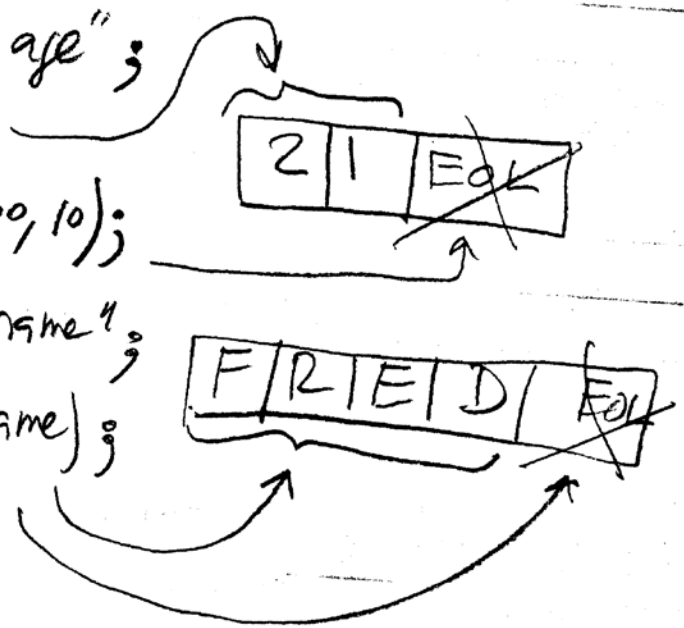
- istream& ignore (streamsize n = 1, int delim = EOF);
- Extracts characters from the input sequence and discards them.
- The extraction ends when *n* characters have been extracted and discarded or when the character *delim* is found, whichever comes first.
- In the latter case, the *delim* character itself is also extracted.
- **Parameters**
  - n -- Maximum number of characters to extract (and ignore). This is an integer value of type streamsize.
  - delim -- Delimiting character.

Why cin.ignore is so important

```
cout << "enter age";
cin >> age;
cout << "enter name";
getline (cin, name);
```




```
cout << "enter age";
cin >> age;
cin.ignore(1000, 10);
cout << "enter name";
getline(cin, name);
```



itsAboutYou.cpp example

itsAboutYou.cpp example

- There are four separate examples of keyboard input -- one for an integer, one for a floating point number, and two for text.
- They are entirely independent of each other, and can be used in any combination to match the needs of your program.
- Each has three parts: (1) variable declaration, (2) user prompt, and (3) transfer of data from the keyboard to the memory reserved by the variable.
- The variable declaration does not have to be immediately before the prompt -- it just has to be somewhere before the transfer statement.
- The prompt (with the **cout <<**) alerts the user that the computer is waiting for something to be typed on the keyboard and the ENTER key pressed.
- Note that the prompts are enclosed in quotes (so it's hard to have a quote as part of the prompt itself!)
- Prompts are not required, but it's hard for a user to know what to do without them!

How to capture a string

Keyboard buffer, after pressing enter

string s;

cin>>s;

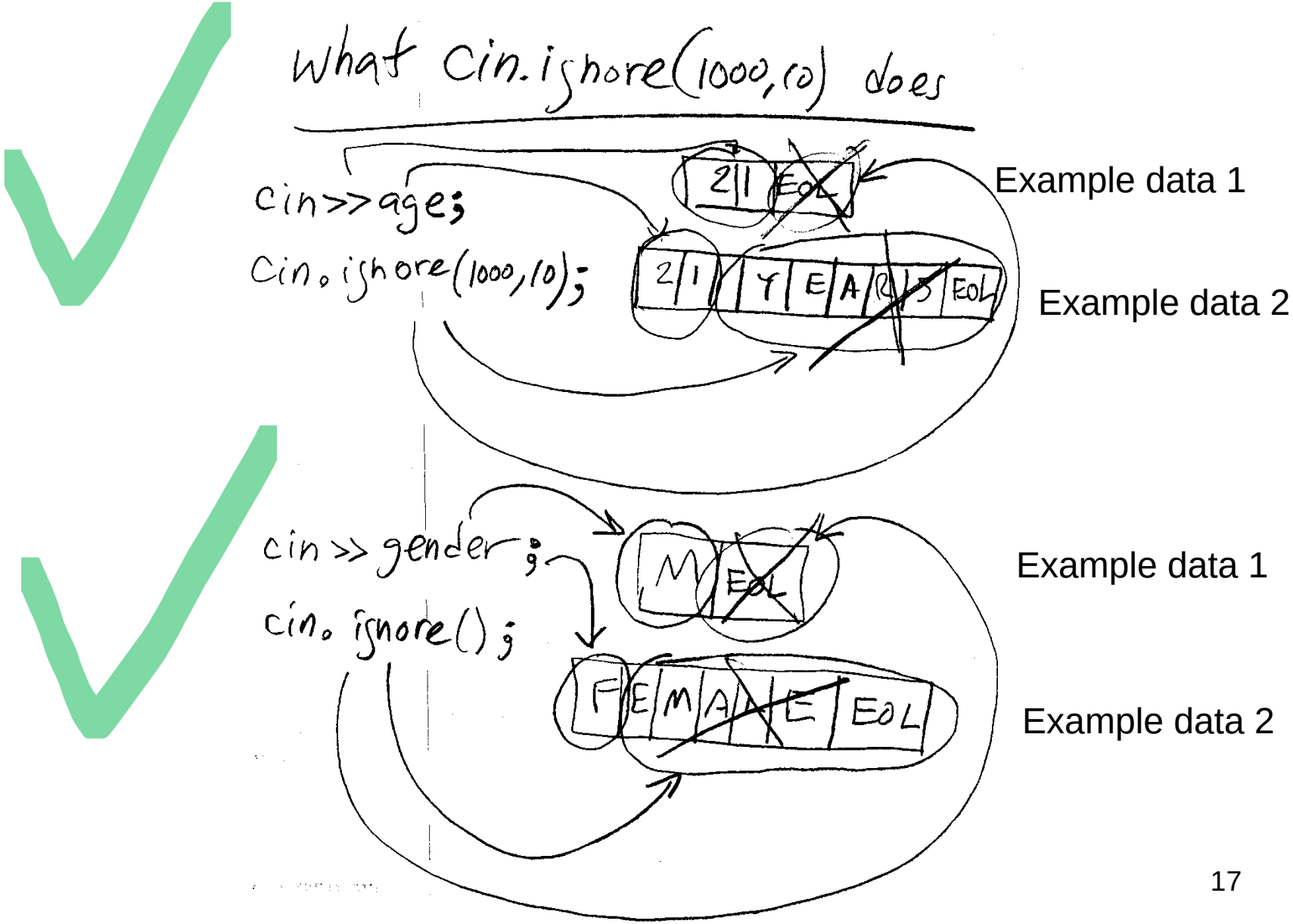
leaves this in buffer

C	O	M	S	C	I	I	O	I	S	F	U	N	EOL	

getline(cin, s);

removes this from buffer

Capturing characters and strings



Transfer statement

- The way the transfer statement is written depends on the data type of the variable being read.
- For numbers and single-character text, two statements are used -- one with **cin >> ...;** , followed by **cin.ignore(1000, 10);** .
- But for text of any length, a single statement is used: **getline(cin, ...); where ...** stands for the variable name.

Prompts

- The statements with **cin** in them cause the program to pause and wait for the user to press the ENTER key before continuing.
- Program execution actually suspends there, waiting for the ENTER key to be pressed.
- Anything that is typed before the ENTER key gets pressed is then captured by the program, and can be stored in a variable.
- But on the computer screen, the only evidence of the **cin** statement is a flashing symbol – usually a horizontal line.
- This is not very much of a clue for the user to know that something is supposed to be typed now, followed by the ENTER key.
- So **cin** is usually preceded by a **cout** , with instructions for the user.
- This is called a "prompt".

With no prompt

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

With same-line prompt

```
string name;
cout << "Enter name: ";
getline(cin, name);
```

```
Enter name: _
```

With a next-line prompt

```
string name;
cout << "Enter name:" <<
endl;
getline(cin, name);
```

```
Enter name:
_
```

Another console input example

- This example presents and solves a formula from a once-secret Los Alamos document, on the predicted radiation dose from a nuclear explosion.
- The input is the strength of the bomb in kilotons, and the distance from the blast in yards (but we will input the distance in miles).
- The result is the #of units of radiation exposure: 25 is insignificant, and 1000 is fatal.
- In between these is varying degrees of health effects.

Using programmer input

```
...
//Libraries
#include <iostream>
using namespace std;
#include <cmath>
…
//main program
int main()
{ //introduction
  cout << “This program calculates radiation does from nuclear blast” << endl;
  cout << “Author: Teacher” << endl;

  //data
  double W = 20.0;  //blast strength in kilotons
  double D = 2.0;  //distance from blast in miles
  double DY = D * 1760.0; // convert miles to yards
  double I = (3.2E9 * W) / (DY * DY) * exp(-DY / 360.0); // radiation dose in roentgens

  //output
  cout << "Blast strength: " << W << " kilotons " << endl;
  cout << "Distance from blast: " << D << " miles " << endl;
  cout << "Predicted radiation dose: " << I << " roentgens " << endl;
}//main
```

Using console input

```
…//libraries
#include <iostream>
using namespace std;
#include <cmath>
…
//main program
int main()
{  //introduction
  cout << “This program calculates radiation does from nuclear blast” << endl;
  cout << “Author: Teacher” << endl; …

  //data
  double W; //blast strength in kilotons
  double D;  //distance from the blast in miles
  double DY; //distance from the blast in yards
  double I;  //predicted radiation dose in roentgens
…
  //user input
  cout << "Enter the blast strength, kilotons: ";
  cin >> W;
  cin.ignore(1000, 10);
  cout << "Enter the distance from blast, miles: ";
  cin >> D;
  cin.ignore(1000, 10);

  //calculate radiation dose in roentgens
  DY = D * 1760; // convert miles to yards
  I = (3.2E9 * W) / (DY * DY) * exp(-DY / 360);
  cout << "Predicted radiation dose: " << I << " roentgens " << endl;
}//main
```

Input multiple values with one prompt

```
…
#include <iostream>
using namespace std;
#include <cmath>
…
int main()
{
  //data
  double W; //blast strength
  double D;  //distance in miles
  double DY; //distance in yards
  double I;  //radiation dose in roentgens

  //input 2 values with only one prompt for multiple
  cout << "Enter the blast strength (kT) and distance (mi), space-separated: ";
  cin >> W;
  cin >> D;

  //output results
  DY = D * 1760; // convert miles to yards
  I = (3.2E9 * W) / (DY * DY) * exp(-DY / 360);
  cout << "Predicted radiation dose: " << I << " roentgens " << endl;
}//main
```

More about prompts

- it is common to include possible responses in the prompt, enclosed in square brackets and separated by commas or slashes, like this:

“Do you want to see a movie? [Y/N]: “

as a guide to the user.

- Unless you have a very good reason to do otherwise, *always* have a prompt for *each* user-entered value.
- Every time you expect the user to type something and press ENTER, or just to press ENTER after a pause, do this: precede it with a same-line or next-line prompt, so that the user knows to do something.
- And if you can include some guidance, such as possible responses or an example response, do so.

Interrupting an interactive program

- Sometimes in the process of writing and testing a program that has console input, it may be convenient to terminate (that is, end) a program early.
- For example, if the input involves a series of prompts, you may notice incorrect spelling in the first prompt. In order to stop right away and fix it, rather than continue on through the remaining prompts and normal termination of the program, you can type CTRL-C to make the program stop.
- Hold down the CTRL key, press the C key, and release both.

Input: Keyboard for Width and Height
Output: Console of Area

```
//libraries
#include <iostream>
using namespace std;
…
int main()
{
  //data
  int width;  //width of a room
  int height; //height of room
  int area;  //calculated area of a room

  // get width and height from the user via the keyboard
  cout << "Enter width in feet (whole numbers only): ";
  cin >> width;
  cin.ignore(1000, 10);
  cout << "Enter height in feet (whole numbers only): ";
  cin >> height;
  cin.ignore(1000, 10);

  // calculate and print the area to the console
  area = width * height;
  cout << "The area is " << area << " square feet" << endl;
}//main
```

References

- INTRODUCTION TO PROGRAMMING USING C++ by Robert Burns