PIC 10A

Lecture 2: Variables, Data types, I/O

Data types

In C++ programs we will deal with different types of data, such as integers, doubles (also called floating-point numbers) as well as strings and booleans.

There are many reasons to “label” data with types:

- To reduce programming mistakes
- Make internal computations easier
- To make our programs clearer

Numeric data types

The two data types that we use in this class to describe numbers are **int** and **double**.

int

Stands for integer

double

double numbers are decimal numbers

Variables

Variable in C++ means something else than a variable in mathematics.

In mathematics variables are most often something whose value is unknown and we are trying to solve.

In C++ variable is an object that is storing a value.

To create a variable we must know what its data type is.

```
int x=2;
```

The above line creates a variable called x which stores data of type integer.

Note that the variable is on the left and the value being assigned to the variable is on the right.

What can I do with a variable?

Lots of cool stuff!!!

In many ways a variable behaves exactly like the data it is storing:

```
int x=2;
cout << x; // outputs 2
```

Notice how there are no quotes around the variable!

How about

```
cout << x+7;
```

The computer will recognize x+7 as an artihmetic expression and evaluate it to 9.

Notice that we are not changing the value of x here!

Book example

```
#include <iostream>
using namespace std;
int main()
{
   int pennies = 8;
   int dimes = 4;
   int quarters = 3;

   double total = pennies * 0.01 + dimes * 0.10
                  + quarters * 0.25;

   cout << "Total value of the coins is: " << total
        << "\n";

   return 0;
}
```

Why not always use doubles?

We could have made all our variables doubles. Why didn't we?

We knew that number of coins is always an integer, so we will catch errors by insisting that they are integers.

We also make it clear to anyone reading the code that number of coins is a whole number.

Integers are faster to process than doubles. This may be significant factor in very large programs.

What happens if we make the variable total an integer?

Choose your data type

You should think carefully about what type of data your variable will store.

```
int x=25.7;  // What happens?


double y=20; // Is this an error?
```

It is good for to make clear what variables are doubles and what variables are ints

so

```
double y=20.0; // good practice
```

Variable names

Look at this code:

```
#include <iostream>
using namespace std;
int main()
{
    int x = 8;
    int y = 4;
    int z = 3;

    double w = x * 0.01 + y * 0.10 + z * 0.25;

    return 0;
}
```

What does this code calculate?

Good practices for variable names

Variable names should be descriptive!

```
a=17; // bad variable name

areatriangle = 17; // slightly better

area_triangle = 17;  //good!!
```

Variable names should be short

Do not do:

```
area_of_a_triangle = 17; // bad!
```

Usually start variables names with lower-case letter.
Generally capitalize class names.
Write constants in all caps. (e.g. PI)

Variable names

Variable names are case sensitive

If you declare a variable

double perimeter;

Do not start using Perimeter in your code. You will get an error.

Variable names must start with a letter or an underscore _ and the remaining characters must be letters numbers or underscores.

You cannot use a reserved word for your variable name like main or cout.

Syntax for variable definitions

```
type_name variable_name;

// or

type_name variable_name=value;

// or

type_name variable_name1=value1, variable_name2=value2, … ;

// or

type_name variable_name1, variable_name2, … ;
```

Examples

Examples:

```
double total=15;

int whole_number;

double side1, side2, area;

double side1=.5, side2=15.4;
```

What is wrong with these:

```
double pic 10a;

double No_Swearing_#$%^!!;

int 50_cents;
```

Declaring Variables

Declare Variables Right Away

Standard to declare all variables first.

```
int main () {
int MyNumber ;
double gravity = 9.8;
**STATEMENTS**
return 0;
}
```

As you need variables in your program, go back to the beginning and declare them.

In some cases people will declare a variable in the middle of a code. Mostly when the variable is to be used only once at that exact spot for some relatively unimportant purpose.

Comments about comments

2 ways to comment on your code:

For a single line comment

```
// This is a comment
```

Use it either on a line of its own:

```
//Variables used to define the dimensions of a rectangle
double width=2.0;
double height=3.7;
```

Or to explain a single line of code

```
cout << Total;  // Here we output the total cost.
```

Comments

```
/* This is another comment method.  Use
this method when you want to make longer comments that span
many lines
*/
```

The top of every source file should contain your name, the date, and a very brief description. Good business practice.

Asterix make your comment stand out

```
/*********************************************************
** hw0.cpp
** Prints "Hello Middle Earth!" on the screen.
** Joe Bruin 30/3/11
** SID 5687384930
*********************************************************/
# include <iostream>
```

Comments

Why use comments?

- Comments are ignored by the computer
- They are for your benefit!
- Comments make coding easier and code more readable
- Other people can read your code and understand it
- Useful for debugging!

Getting input

We already know how to print stuff to the console

```
double side1 = 5.6;
double side2 = 4.5;

cout << "The Area of the rectangle is: " << side1 *
side2<< "\n";
cout << "The perimeter of the rectangle is: " << 2*
side1 + 2*side2;
```

How do we get input?

Use cin!

```
int number;
cin >> number;
cout << "Your number is: " << number;
```

cin

cin means that we are getting data from the console. (We could get data from other places too...More on that later in quarter.)

Note that our “arrows” >> are pointing into the page because we are pulling data in.

```
double side1, side2;

cout << “Please give me lengths of two sides of a
rectangle.\n”;
cin >> side1 >> side2; // cin can also be chained
cout << “The area is: “ << side1 * side2;
```

`cin` buffer

When too many values are entered by the user they get placed into a buffer. Buffer is like a temporary storage place.

When the computer encounters a cin command it first checks to see if anything is in the buffer.

If there is something in the buffer the input for the variable is read from the buffer instead of the console!

```
int dimes;
int quarters;
cout << "Please enter number of dimes.";
cin >> dimes; // If a user enters two numbers here say 15 20
cout << "Please enter number of quarters.";
cin >> quarters; // User does not get a chance to input a value
                 //  Instead quarters gets a value 20
```

Be careful with data types

```
int x,y;
cin >> x; // What happens if user types in 3.14?
cin >> y;
cout << x << " " << y; // What is the output?
```

`cin` fail state

When cin tries to read incompatible data to a variable, cin goes into a fail state.

This means that cin has lost confidence in the data in the buffer.

All subsequent cin commands will be ignored!!

There is a way to recover from this error. We will learn later how.

Users always find a way to do something incorrect.

Checking input for validity is important. We will learn how to do this later.

For now we will assume that the user always enters correct data.

More about data types

All data is internally represented by 1's and 0's.

• Numbers are represented in binary as strings of 1's and 0's, called bits.

• 2 = 00000010
• 131 = 01000011

We can also represent decimals. The "longer" the number, the more bits it takes to represent.

So number 0.0002 takes more bits to store than the number 2.

• 8 bits = 1 byte 1,000 bytes = 1 kilobyte
• 1,000,000 bytes = 1 megabyte

More data types

Type	Description	Number of bits
bool	boolean (true/false) 1 or 0	1
char	character	8
short	short integer	16
int	integer eg. -17, 0, 256	32
float	floating point 3.14159	32
double	long float	64
long double	very long float	80