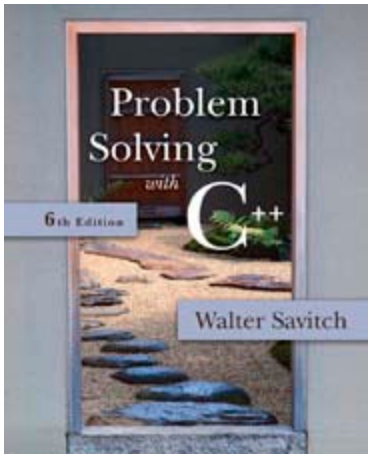
APS105: Lecture 6

Wael Aboelsaadat

wael@cs.toronto.edu

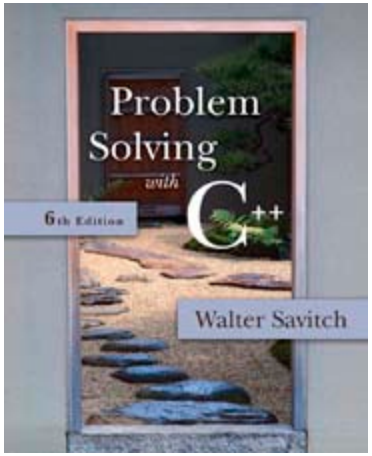
http://ccnet3.utoronto.ca/20079/aps105h1f/

Acknowledgement: These slides are a modified version of the text book slides as supplied by Addison Wesley

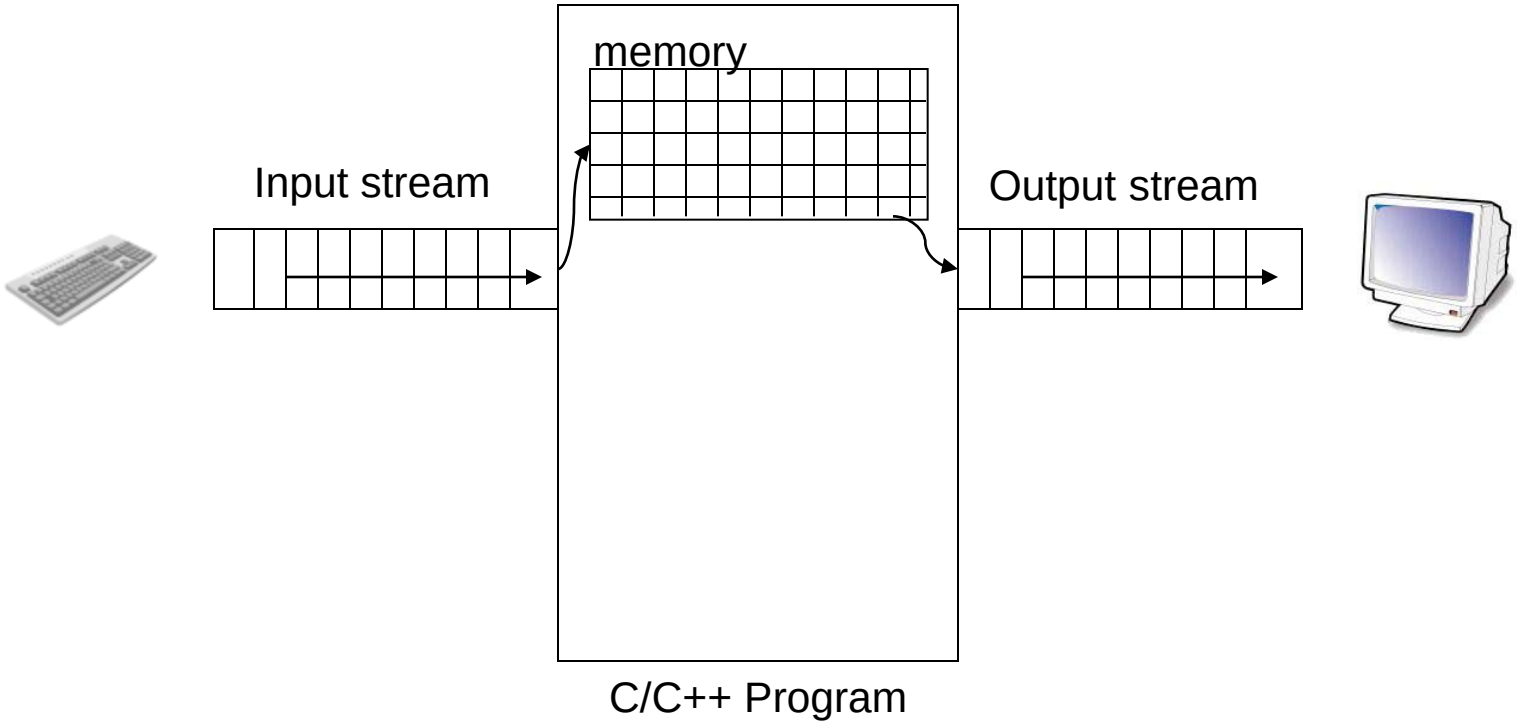


2.2

Input and Output



Input and Output



Examples Using cout

- This produces the same result as the previous sample

```
cout << number_of_bars ;
cout << " candy bars\n";
```

- Here arithmetic is performed in the cout statement

```
cout << "Total cost is $" << (price + tax);
```

- Quoted strings are enclosed in double quotes ("Walter")
  - Don't use two single quotes (')
- A blank space can also be inserted with

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

if there are no strings in which a space is desired as in " candy bars\n"

Reading Data From cin

- Multiple data items are separated by spaces
- Data is not read until the enter key is pressed
  - Allows user to make corrections

- Example:

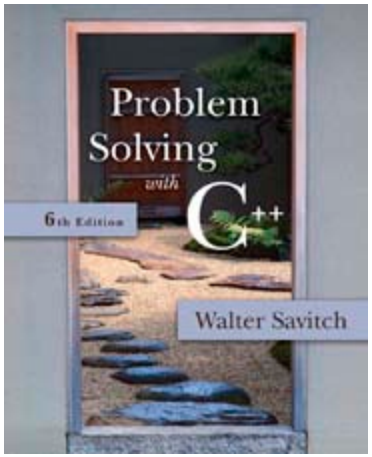
```
cin >> v1 >> v2 >> v3;
```

  - Requires three space separated values
  - User might type

```
34  45  12   <enter key>
```

2.3

Data Types and Expressions



Other Number Types

- Various number types have different memory requirements
  - More precision requires more bytes of memory
  - Very large numbers require more bytes of memory
  - Very small numbers require more bytes of memory

Display 2.2

Integer types

- long or long int (often 4 bytes)
  - Equivalent forms to declare very large integers

```
long  big_total;
long int big_total;
```

- short or short int (often 2 bytes)
  - Equivalent forms to declare smaller integers

```
short small_total;
short int small_total;
```

Floating point types

- long double (often 10 bytes)
  - Declares floating point numbers with up to 19 significant digits

```
long double big_number;
```

- float (often 4 bytes)
  - Declares floating point numbers with up to 7 significant digits

```
float not_so_big_number;
```

Type char

- Computers process character data too
- char
  - Short for character
  - Can be any single character from the keyboard

- To declare a variable of type char:
-

```
char letter;
```

char constants

- Character constants are enclosed in single quotes

```
char letter = 'a';
```

Reading Character Data

- cin skips blanks and line breaks looking for data
- The following reads two characters but skips any space that might be between

```
char symbol1, symbol2;
cin  >>  symbol1  >>  symbol2;
```

- User normally separate data items by spaces

```
J  D
```

- Results are the same if the data is not separated by spaces

```
JD
```

Display 2.3

Type string

- Complex/compound type
- Library to do string processing
- Strings of characters, even if only one character is enclosed in double quotes
  - "a" is a string of characters containing one character
  - 'a' is a value of type character

Display 2.4

Type bool

- bool is a new addition to C++
  - Short for boolean
  - Boolean values are either true or false

- To declare a variable of type bool:

```
bool old_enough;
```

Type Compatibilities

- In general store values in variables of the same type
  - This is a type mismatch:

```
int int_variable;
int_variable = 2.99;
```

  - If your compiler allows this, int_variable will most likely contain the value 2, not 2.99

int ←→ double (part 1)

- Variables of type double should not be assigned to variables of type int

```
int int_variable;
double double_variable;
double_variable = 2.00;
 int_variable = double_variable;
```

  - If allowed, int_variable contains 2, not 2.00

int ←→ double (part 2)

- Integer values can normally be stored in variables of type double

```
double double_variable;
double_variable = 2;
```

  - double_variable will contain 2.0

char ← → int

- The following actions are possible but generally not recommended!
- It is possible to store char values in integer variables

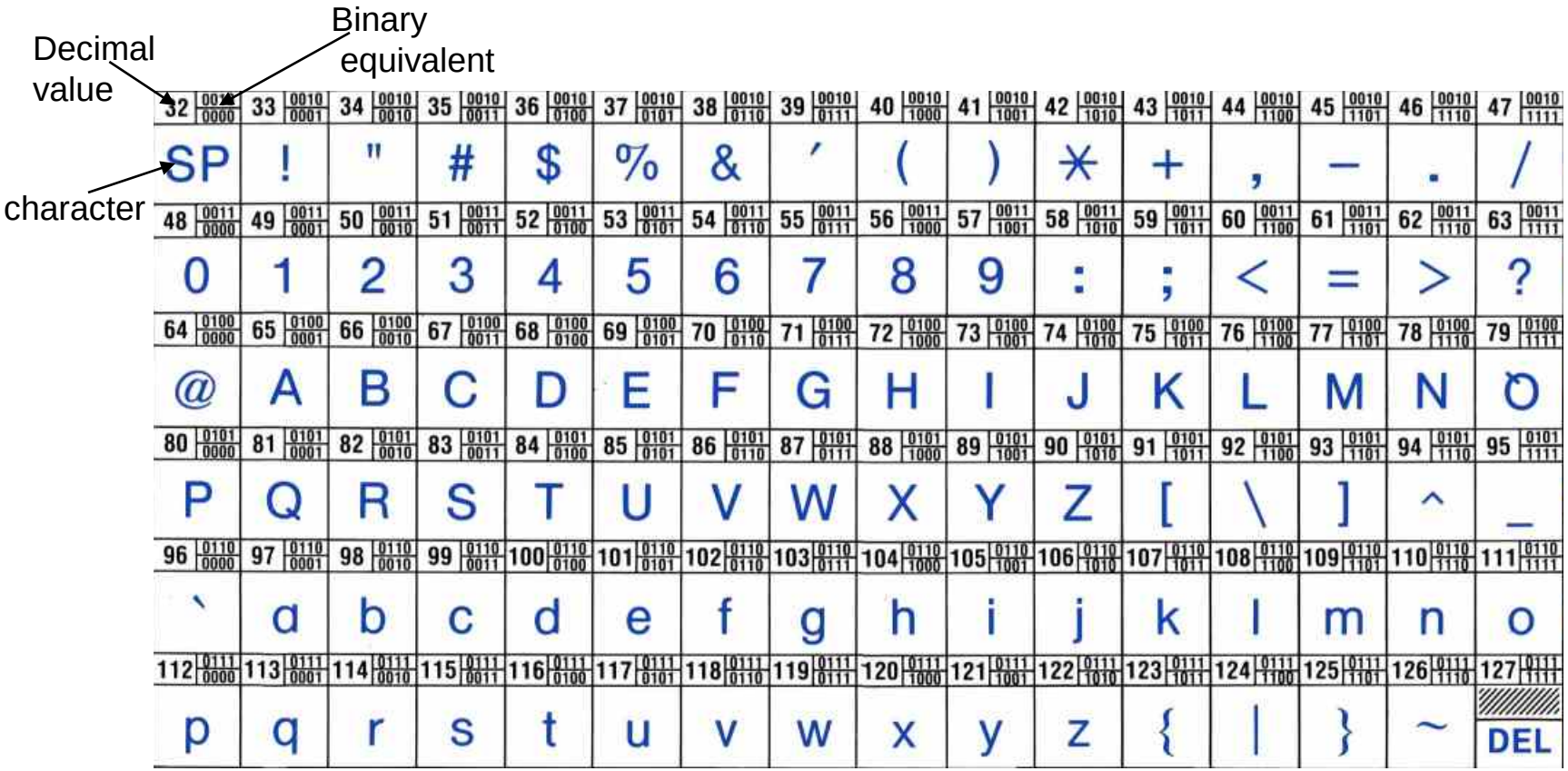
  int value = 'A';

  value will contain an integer representing 'A'

- It is possible to store int values in char variables

  char letter = 65;

char ← → int, Why it works?



32 0010 0000	33 0010 0001	34 0010 0010	35 0010 0011	36 0010 0100	37 0010 0101	38 0010 0110	39 0010 0111	40 0010 1000	41 0010 1001	42 0010 1010	43 0010 1011	44 0010 1100	45 0010 1101	46 0010 1110	47 0010 1111
SP	!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
48 0011 0000	49 0011 0001	50 0011 0010	51 0011 0011	52 0011 0100	53 0011 0101	54 0011 0110	55 0011 0111	56 0011 1000	57 0011 1001	58 0011 1010	59 0011 1011	60 0011 1100	61 0011 1101	62 0011 1110	63 0011 1111
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64 0100 0000	65 0100 0001	66 0100 0010	67 0100 0011	68 0100 0100	69 0100 0101	70 0100 0110	71 0100 0111	72 0100 1000	73 0100 1001	74 0100 1010	75 0100 1011	76 0100 1100	77 0100 1101	78 0100 1110	79 0100 1111
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80 0101 0000	81 0101 0001	82 0101 0010	83 0101 0011	84 0101 0100	85 0101 0101	86 0101 0110	87 0101 0111	88 0101 1000	89 0101 1001	90 0101 1010	91 0101 1011	92 0101 1100	93 0101 1101	94 0101 1110	95 0101 1111
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96 0110 0000	97 0110 0001	98 0110 0010	99 0110 0011	100 0110 0100	101 0110 0101	102 0110 0110	103 0110 0111	104 0110 1000	105 0110 1001	106 0110 1010	107 0110 1011	108 0110 1100	109 0110 1101	110 0110 1110	111 0110 1111
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112 0111 0000	113 0111 0001	114 0111 0010	115 0111 0011	116 0111 0100	117 0111 0101	118 0111 0110	119 0111 0111	120 0111 1000	121 0111 1001	122 0111 1010	123 0111 1011	124 0111 1100	125 0111 1101	126 0111 1110	127 0111 1111
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

bool ← → int

- The following actions are possible but generally not recommended!
- Values of type bool can be assigned to int variables
  - True is stored as 1
  - False is stored as 0
- Values of type int can be assigned to bool variables
  - Any non-zero integer is stored as true
  - Zero is stored as false

Results of Operators

- Arithmetic operators can be used with any numeric type (+, - , * , /, %)
- An operand is a number or variable used by the operator
- Result of an operator depends on the types of operands
  - If both operands are int, the result is int
  - If one or both operands are double, the result is double

Division of Doubles

- Division with at least one operator of type double produces the expected results.

```
double divisor, dividend, quotient;
divisor = 3;
dividend = 5;
quotient = dividend / divisor;
```

  - quotient = 1.6666…
  - Result is the same if either dividend or divisor is of type int

Division of Integers

- Be careful with the division operator!
  - int / int produces an integer result
    (true for variables or numeric constants)

```
int dividend, divisor, quotient;
dividend = 5;
divisor = 3;
quotient = dividend / divisor;
```

  - The value of quotient is 1, not 1.666…
  - Integer division does not round the result, the fractional part is discarded!

Arithmetic Expressions

- Use spacing to make expressions readable
  - Which is easier to read?

Display 2.5

x+y*z or x + y * z

- Precedence rules for operators are the same as used in your algebra classes
- Use parentheses to alter the order of operations

  x + y * z (y is multiplied by z first)

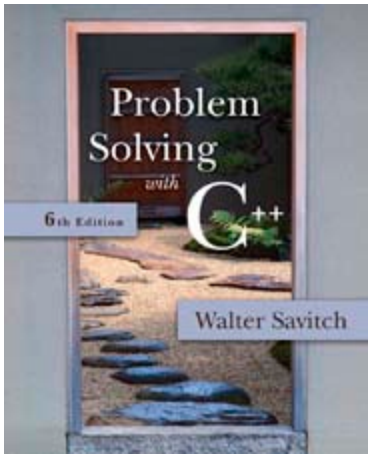
  (x + y) * z (x and y are added first)

Operator Shorthand

- Some expressions occur so often that C++ contains to shorthand operators for them
- All arithmetic operators can be used this way
  - += count = count + 2; becomes
    count += 2;
  - *= bonus = bonus * 2; becomes
    bonus *= 2;
  - /= time = time / rush_factor; becomes
    time /= rush_factor;
  - %= remainder = remainder % (cnt1+ cnt2); becomes
    remainder %= (cnt1 + cnt2);

2.4

Simple Flow of Control



Simple Flow of Control

- Flow of control
  - The order in which statements are executed
- Branch
  - Lets program choose between two alternatives

Branch Example

- To calculate hourly wages there are two choices
  - Regular time (up to 40 hours)
    - gross_pay = rate * hours;
  - Overtime (over 40 hours)
    - gross_pay = rate * 40 + 1.5 * rate * (hours - 40);
  - The program must choose which of these expressions to use

Designing the Branch

- Decide if (hours >40) is true
  - If it is true, then use

    gross_pay = rate * 40 + 1.5 * rate * (hours - 40);

  - If it is not true, then use

    gross_pay = rate * hours;

Implementing the Branch

- if-else statement is used in C++ to perform a branch

  - if (hours > 40)
    gross_pay = rate * 40 + 1.5 * rate * (hours - 40);
  - else
  - gross_pay = rate * hours;

Display 2.7

Display 2.2

DISPLAY 2.2 Some Number Types

Type Name	Memory Used	Size Range	Precision
short (also called *short int*)	2 bytes	–32,767 to 32,767	(not applicable)
int	4 bytes	–2,147,483,647 to 2,147,483,647	(not applicable)
long (also called *long int*)	4 bytes	–2,147,483,647 to 2,147,483,647	(not applicable)
float	4 bytes	approximately 10^{-38} to 10^{38}	7 digits
double	8 bytes	approximately 10^{-308} to 10^{308}	15 digits
long double	10 bytes	approximately 10^{-4932} to 10^{4932}	19 digits

These are only sample values to give you a general idea of how the types differ. The values for any of these entries may be different on your system. *Precision* refers to the number of meaningful digits, including digits in front of the decimal point. The ranges for the types *float*, *double*, and *long double* are the ranges for positive numbers. Negative numbers have a similar range, but with a negative sign in front of each number.

Display 2.3


```
#include <iostream>
using namespace std;
int main( )
{
    char symbol1, symbol2, symbol3;

    cout << "Enter two initials, without any periods:\n";
    cin >> symbol1 >> symbol2;

    cout << "The two initials are:\n";
    cout << symbol1 << symbol2 << endl;

    cout << "Once more with a space:\n";
    symbol3 = ' ';
    cout << symbol1 << symbol3 << symbol2 << endl;

    cout << "That's all.";

    return 0;
}
```

Sample Dialogue

```
Enter two initials, without any periods:
J B
The two initials are:
JB
Once more with a space:
J B
That's all.
```

Display 2.4

Back Next

DISPLAY 2.4 **The `string` class**

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main()
5  {
6      string middle_name, pet_name;
7      string alter_ego_name;
8
9      cout << "Enter your middle name and the name of your pet.\n";
10     cin >> middle_name;
11     cin >> pet_name;
12
13     alter_ego_name = pet_name + " " + middle_name;
14
15     cout << "The name of your alter ego is ";
16     cout << alter_ego_name << "." << endl;
17
18     return 0;
19
20 {
```

Sample Dialogue 1

```
Enter your middle name and the name of your pet.
Parker Pippen
The name of your alter ego is Pippen Parker.
```

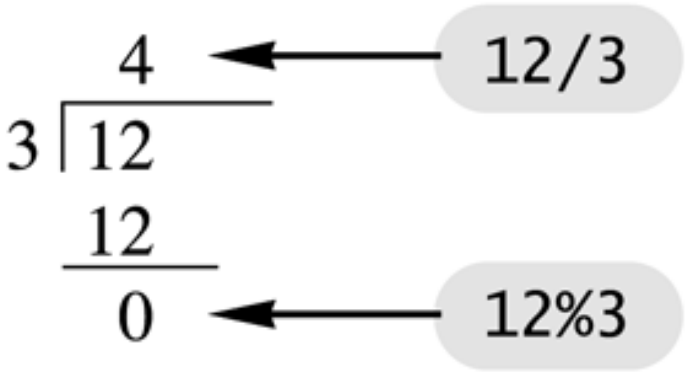
Sample Dialogue 2

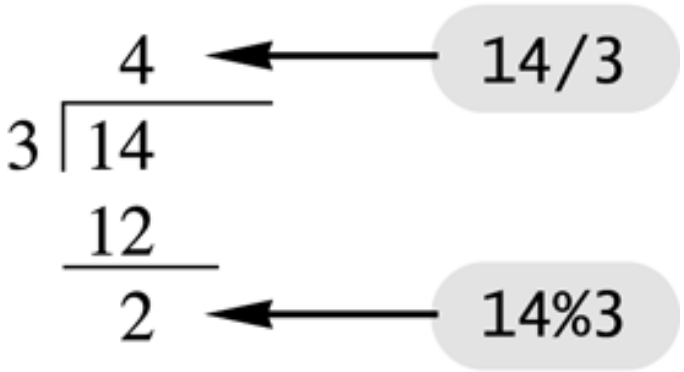
```
Enter your middle name and the name of your pet.
Parker
Mr. Bojangles
The name of your alter ego is Mr. Parker.
```

Display 2.5

Back Next

Integer Division





Display 2.6

Back Next

Arithmetic Expressions

Mathematical Formula	C++ Expression
$b^2 - 4ac$	`b*b - 4*a*c`
$x(y + z)$	`x*(y + z)`
$\frac{1}{x^2 + x + 3}$	`1/(x*x + x + 3)`
$\frac{a + b}{c - d}$	`(a + b)/(c - d)`

Display 2.7

An *if-else* Statement

```
#include <iostream>
using namespace std;
int main( )
{
    int hours;
    double gross_pay, rate;

    cout << "Enter the hourly rate of pay: $";
    cin >> rate;
    cout << "Enter the number of hours worked,\n"
         << "rounded to a whole number of hours: ";
    cin >> hours;

    if (hours > 40)
        gross_pay = rate*40 + 1.5*rate*(hours - 40);
    else
        gross_pay = rate*hours;

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "Hours = " << hours << endl;
    cout << "Hourly pay rate = $" << rate << endl;
    cout << "Gross pay = $" << gross_pay << endl;

    return 0;
}
```

Sample Dialogue 1

```
Enter the hourly rate of pay: $20.00
Enter the number of hours worked,
rounded to a whole number of hours: 30
Hours = 30
Hourly pay rate = $20.00
Gross pay = $600.00
```

Sample Dialogue 2

```
Enter the hourly rate of pay: $10.00
Enter the number of hours worked,
rounded to a whole number of hours: 41
Hours = 41
Hourly pay rate = $10.00
Gross pay = $415.00
```