CHPT3

cin ===== can be used to read data typed at keyboard

must have : #include<iostream> for program to use cin cout

prompt ===== tell the user what they need to do

>> stream extraction operator ===== extracts characters from the input stream

Two steps: 1. Use cout display prompt on screen

2. Use cin to read a value form the keyboard

Input buffer ==== keyboard buffer

User enter characters from keyboard, they are temporarily in ==== input buffer ====keyboard buffer

Entering multiple values

cin >> length >> width

spaces between length and width [Enter] after the the last number

different data types can be brought in with cin>> length >> width>> letter;

expression === a programming statement that has a value ==== has operands operator

OPERATORS Associativity

operator Precedence ( ) - negation unary right to left

\* / % binary left to right

+ - minus binary left to right

Grouping with parentheses

Algebraic expression to programming statement

pow(base, power) ===== argument: base power

return value === answer

first argument ==== double

return =====double

#include <cmath> header file need for pow

++++++++++++++++++++++++++++++++++++++++++++++++++++

Data Type ranking long highest ==== can hold a largest number

double

float

unsigned long

long

unsigned int

int lowest ==== hold the smallest number

Three rules: DATA RANKING

1. Char, short and unsigned short values are automatically promoted to int values
2. When operator works with two values of different data types, the lower ranking value is promoted to type of higher-ranking value
3. When the final value of an expression is assigned to a variable, it will be converted to the data type of that variable.

Type Casting

Type cast expression ==== manually promote or demote a value

Static\_cast<DataType>(value)

booksPerMonth = static\_cast<double>(books)/ months;

integer division ===== prevent when one of operand is converted to a double before

division operation.

Forces the c++ to convert the other operand value to double

c-style and Prestandard C++ type cast Expression

c-style prefix notation (int)books/ months

prestandard C++ functional notation double(books)/ months

standard C++ static\_cast <static\_cast<int>(books)/months

++++++++++++++++++++++++++++++++++++++++++++++++++

Overflow

Value cannot fit in the number of bits provided by a variable’s data type:

Overflow : 32768 in 2 bytes largest

High-order bit is sign bit

Thus if add 1 bit

Turn to -0 bcs of overflow

Underflow: 3.0E-47 Smallest

Constant variable: name is all CAPS ==== read-only

const double INTEREST\_RATE = .068;

a variable

cannot change value while program is running

#define PI 3.14159 C-style

Text substitution before compiler

NO semicolon

Multiple and combined Assignment

Combined assignment operators == compound operators === arithmetic assignment operators

+= -= \*= /= %=

Formatting output

Formatting === the way a value is printed

setw(2) === field width ==== right-justified

==== must have #include<iomanip> to use setw(2)

==== count decimal point

1. Floating point number=== includes position for decimal point
2. String === includes all characters and spaces
3. Right- justified by default

setprecision(5) ==== significant digits ==== precision

4.91877 system: float-point default 6 significant digits

setprecision(5) 4.9188 5 significant digits ===rounded

==== must have #include<iomanip> to use setprecision(5)

=== no trailing 0 ==== round if not enough space

Fixed-point ===== fixed

cout<< fixed << setprecision(2)

\*\*\*\*setprecision(2) === use with fixed ====

mean two digits\*\*\* after the decimal\*\*\*\*

showpoint ====

OVERRIDE the defaults of float-point numbers:

1. Display without trailing zeros
2. Floating point numbers with no fractional parts display without decimal point

Thus: double x = 456.0;

cout<< x<<endl // display 456

cout<< showpoint<< x<<endl; // display 456.000

default 6 sign

cout<<fixed<<showpoint<<setprecision(2)<<x<<endl;

// display 456.00

+++++++++++++++++++++++++++++++++++++++++++++++

Left and right manipulator

Normally right-justified

left left-justified

right right-justified

inputting string

cin>> name ==== stop at space

getline(cin, name); ===== get spaces ==== stop at [enter]

cin.get() member function

cin.get(ch); == is same as ==== ch = cin.get();

cin.ignore(number,characterC)

will skip a number of characters until it get to character

cin.ignore() will only skip next character

string state = “New Jersey”

state.length () === count ===space ===as part of length

but ===not \0

spaces.assign(22, ’ ‘);

+ operator

C-string char word[10] = “Hello”; pg… 127

strcpy( name2,”Sebastian”);

++++++++++++++++++++++++

Buffer overrun === when the input is bigger than the

char word[5];

cin.width(5);

cin>> word // if word is >5 length

cin.setw(5) //display 4 letters only p129

Overwriting

cin.getline(sentence, size) ===== get one less than size