

Semester project

Final version due 12/11; project is submitted in several preliminary versions throughout semester

You are to write a simple menu-driven inventory-management application as specified below.

1. *Overview*: Your application will allow the user to manage information about one inventory item (*record*, in database terminology) and, later, as a challenge part of the project, a *collection* of items stored in a disk file. You will define what kind of inventory items your application manages.

2. An inventory item has the following attributes:

- An identification number;
- A one-word name (hyphens OK);
- Quantity on hand;
- Cost;
- Some other attribute(s) defined by you.

3. Your application will display a menu of user options. The initial version will have the following choices:

- Input inventory item;
- Display item;
- Update item;
- Save item to disk;
- Retrieve item from disk.

Your application will perform, or allow the user to perform, the operation corresponding to the chosen menu item. Display an error message if user requests output of data before inputting or retrieving data.

4. The *Display* operation will display an inventory item in a tabular format, as below, including the total value of all the instances of the item in stock:

ID	Name	Cost	Qty	On-hand value
--	----	-----	---	-----
31	pencil	.25	40	10.00

5. Initially your project will consist of separate programs, one to display the main menu and one that will execute operations in sequence. Later the separate programs will be integrated.

6. When you integrate your main-menu program with your implementation of the five operations, write a sub-menu under the Update option, allowing the user to choose which attribute of the item to update. Also, all input data should be *validated*. For example, negative input values should result in error messages. A special challenge is to implement the saving of multiple inventory items in your file and to enable searching and display of this file.

7. The code submission for homework 6 should be in *modular* form, with subprogram calls for each menu option and a function definition for each module. Be sure that each subprogram is properly documented at the top of the function definition.

8. In the final project submission, implement your inventory item using a *structure type* or class and modify your code throughout to use this object-based feature. This means defining a data type using *struct* or *class*, whose data members are the attributes of an inventory item, and replacing all groups of variables and parameters that represent these attributes, of type *int*, *double*, and *string*, by instances of the inventory-item structure type or class.

9. An important part of each submission will be *several clearly written paragraphs* that summarize:

- what the program does (specification), i.e., how the program interacts with the user;
- how it works internally (design). This section should include *flowcharts*, *module hierarchy diagrams*, and *UML class diagrams*.

Each submission should update, rather than replace, earlier documentation.

10. *Test results* must show that the code works correctly for all menu options and a variety of input data.

11. Final submission should be separated into sections on *specification*, *design*, *code*, and *testing*. Use dividers in a thin (not loose-leaf) binder. Submit a disk with your source code and a sample data file.

12. A realistic *optional challenge* part of the project is to extend the application to maintain a disk file of *multiple* inventory records. In that version, the application will have the following menu choices:

- New item (append to file);
- Search by ID (if found, display the item and display a menu to allow update or delete the item);
- Display all items in collection, including summary information;
- Generate a report file in same format as used by the display option.

If you wish to work with arrays as well (not part of this course), you may implement a collection of records using an array.

Requirements for graded assignments

- Each submission must include a description of the *process* by which you did the work, such as what difficulties you had, what new skills you learned, what new concepts became clearer, what work if any is not completed, etc. One part of process documentation can be *muddy concepts* feedback, in which you describe what is least clear in the classroom presentation and textbook.
- Code, documentation, and input/output should all be clear and readable; presentation counts.
- Comments should express *clearly* what the program does. Document program's name and purpose at top in a comment; document any function definitions similarly. Place your name, the date, and the assignment and problem numbers at the top of each source file.
- Use a standard indenting scheme.

- Homework *must* be submitted on time for credit. "It is *strongly* recommended that you start very early on each homework assignment and *ask questions* if any part is unclear or seems difficult" (syllabus).
- You are responsible for submitting only your own work and keeping your own work to yourself so that others will be able to work independently. Work with copied code or comments will be returned ungraded.

Grading criteria for final submissions

- | | |
|-------------------------------|-----|
| • Specification documentation | 15% |
| • Design documentation | 15 |
| • Presentation | 10 |
| • Data definition | 15 |
| • Control structures | 15 |
| • Modules | 15 |
| • Testing | 15 |

Notes

All students are to work independently and submit their own code and documentation.

Note that the semester project is only part of the programming that students are expected to do as lab work for CS I. It may be important for your work on the project that you solve ungraded programming problems (see Homeworks 1-7) in order to learn the skills needed to meet the project specifications.

The project is designed to be done in stages. It is not acceptable to skip these stages and submit only the entire project at the end of the semester.

You may choose to add features to the above specification. Be sure to document these.

Homework 0 (due 9/5)

Log in at <http://framingham.blackboard.com>, “enroll” in 63.152 CS I, and respond to the message from instructor at the Discussion Board.

Textbook chapter-review assignment (due date varies)

Each student will write a one or two page typed review of one chapter of the textbook associated with one of the seven topics of the course. Chapters will be assigned the first week. For each student, the date due will be in the middle of the discussion of the topic of the reviewed chapter. (This is to allow use of the chapter reviews to guide the classroom presentation.)

The following are among topics of discussion appropriate for this assignment:

- Skills presented in the chapter
- Main concepts and terms covered
- Concepts still muddy after reading the chapter
- Comparison of chapter with classroom presentation
- Suggestions for improvement
- Chapter exercises (try one)

Homework 1 (Hardware; due 9/18)

Graded:

Use the *Notepad* Windows accessory, or your C++ program editor, to write programs in the assembler language of the model processor to solve problems 1-2. Test your programs and use the printer to list your program (*.asm*) file and sample input/output file (*.out*). Include comments as specified in the syllabus (name and purpose of program, your name, date, assignment number).

1. Accept input of three integers to represent dollar amounts for price, discount, and sales tax. Display the total amount due after the discount, including tax. *Sample I/O:*

```
[Input:] 200  
[Input:] 20  
[Input:] 10  
[Output:] 190
```

2. Modify *abs.asm* (see Appendix A or slide) to input two values and to display the absolute value of the difference between them.

Sample I/O:

```
[Input:] 6  
[Input:] 8  
[Output:] 2
```

3. Solve these problems, showing your work:
 - (a) Convert 10111 to decimal
 - (b) Convert 18 to binary
 - (c) Add 1100 and 0101

#1- 2: Documentation	/ 20
#1- 2: Code	/ 40
#1- 2: Tests	/ 20
#3	/ 20

Ungraded:

4. Horstmann, p. 29, Ex. R1.1.
5. The following program has an error. Modify it so that it meets its specification.

```
// negate.asm  
// Displays arithmetic negation of input  
input n  
load n  
sub n  
sub n  
print n  
n data 0
```
6. (*Suggestion:* Do this before doing #1.) Use the program, *asmwin.exe*, as described in Appendix A and found in subdirectory *01* of your work disk, to run the program *xy.asm* (below) in step mode.

```
input x  
load x  
add x  
add x  
store y  
print y  
stop  
x data 0  
y data 0
```

Test the program for two or more different input values.

- (a) From your observation, what occurs when the fifth line of the program, *store y*, executes?
- (b) Based on your observation, write a formula, to accurately describe the relationship between program input and output of program *xy.asm*. (Your formula could be of the form “Output is *n* larger than input,” or “Output is random,” or “Output is same as input.”)
- (c) Suggest a better name for the program than *xy.asm* and a better name for the data labels *x* and *y*.
- (d) Using a text editor such as Notepad, print a listing of *xy.asm* and *xy.out*, the record of your test of the program. (*Optional:* add a comment with your answer to (b) and rename the program and variables per (c).)

Using the microprocessor simulator **ASM.EXE**

How to copy the simulator to your floppy disk or network account area:

1. At <http://framingham.blackboard.com>, enroll in the virtual course S03.63152.WE, go to "Course Documents," and download to your own hard disk the file, ASM_SETUP.EXE. This is an install file executable under Windows. Another way to obtain ASM_SETUP.EXE is from someone you trust who has it, such as the instructor or another student.

2. Execute this program by double-clicking on its icon. The installer will place a directory on your hard disk containing the processor-simulator software and some associated files. You only need to execute ASM_SETUP.EXE once per computer.

How to run the simulator:

Once you have installed ASM.EXE, you may run it as often as you wish by clicking on the multicolored icon. Follow instructions on the screen and in Appendix A of the Keil/Johnson text.

How to use Windows Notepad to edit text files

Notepad is a text editor built into the Windows operating environment. You may use it to create or modify text files, as opposed to files specially associated with applications like word processors or spreadsheets. Program source code, including assembler-language and C/C++ code, is normally in the form of text files.

To run Notepad:

Choose *Start / Programs / Accessories / Notepad*.

To enter new text:

Type at the keyboard.

To edit an existing text file:

Choose *File / Open* and navigate the directory.

To save a file:

1. To save a new file or to save an existing file under a different name from before, choose *File / Save As*. Enter the name of your file, either with the extension *.txt* or with a file name and extension both in double quotes.

2. *Pitfall:* If you omit the double quotes, your file will be saved with a second extension, *.txt*. (Often extensions are invisible in the Windows directory system, so that a file named *prog1.asm.txt* will appear as *prog1.asm*.)

To print a file

With the file in the edit area, choose *File / Print*.

Homework 2 (Program design; due 10/1)

Graded:

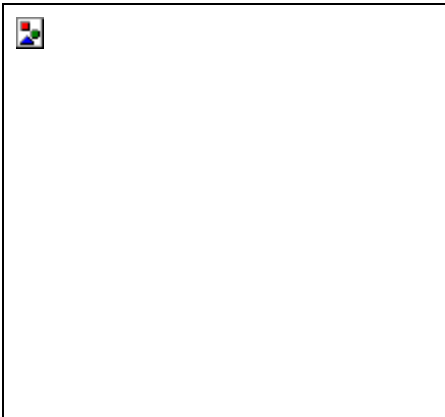
See semester project specification sheet.

- Use your own words to explain the specifications of the project; specifically, what are the inputs, what are the outputs, and what is the relationship of inputs to outputs.
- Write a flowchart of the main-menu loop described in the specification.
- Write a module hierarchy for the application. Describe the modules needed to implement the specifications (#1 above), considering the menu items as modules.

1 (spec):	/ 40
2 (flowchart):	/ 30
3 (module hier.):	/ 30

Ungraded:

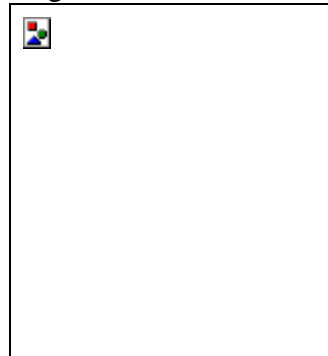
- Horstmann, p. 30, Ex. R1.16
-



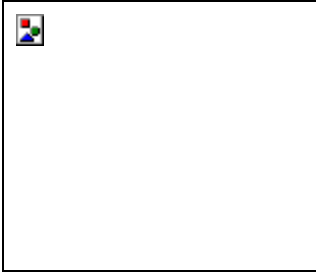
Use the table below to trace the algorithm specified in the flowchart above, assuming input of 2, 3, and 1 in that order.

<i>term</i>	<i>count</i>	<i>total</i>	<i>output</i>
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

- Write structured pseudocode or a structured flowchart for an algorithm to solve *one* of the following problems:
 - Prompt for two integers and display their product, using only addition and subtraction operations in your calculations.
 - Prompt for two integers and display their integer quotient and the remainder, using only subtraction and addition.
 - Prompt for an integer and display “even” if it is divisible by two, “odd” otherwise.
 - Input integers until the current input is less than the previous input.
 - Modify *count.asm* to count up from 1 to 10.
 Output: 12345678910
- Using the assembler language of the micro-processor simulator *asm*, write a program to solve one of the problems in #5 above. Submit printed listings and test results with documentation.
- Use the language of ASMWIN to solve one of these problems.
 - Input two integers and display the larger one.
 - Display each of the *even* integers from 1 to 20.
 - Using *jump0* or *jump-*, loop to input exactly 6 integers and display the largest and smallest.
 - Display the logarithm of an input value, to the base 2. (Below is a flowchart to solve the problem. If logarithm of n is 4, then $2^4 = n$.)



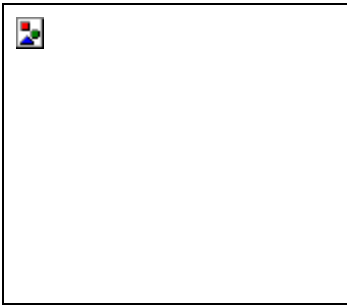
9.



Consider the flowchart above. For each of the input pairs (a, b) shown below, show the resulting output:

<i>Input a</i>	<i>Input b</i>	<i>Output</i>
1	4	_____
2	0	_____
3	3	_____

10.



Consider the flowchart above. For each of the input pairs (a, b) , shown below, show the resulting output:

<i>Input a</i>	<i>Input b</i>	<i>Output</i>
2	1	_____
1	3	_____
4	2	_____

Informally, what does the output of this algorithm tell about the two inputs?

Homework 3 (C and C++ basics; due 10/10)

Graded:

As specified on the course syllabus, document all code with the name and purpose of the program, your name, the date, and the homework number; document process. Use meaningful variable names.

1. Write a program that displays a menu, as indicated on the semester project specification sheet. Menu should display a number or letter beside each menu option, input into a variable a number or letter corresponding to the user's choice, and echo (display) the choice token. (E.g., "You chose 1.")
2. Write a second program that declares integer variables corresponding to the *numeric* attributes of an inventory item; prompts for and inputs these values from the keyboard; and displays these values with labels indicating what they stand for.

Documentation	/ 25
Code	/ 50
Tests	/ 25

Ungraded:

3. Horstmann, p. 32, Ex. P1.4
4. your compiler, compile, link and execute program *hello.cpp* (in directory 03), so that an executable machine language version of the program is written to your network account or work disk. Then compare the sizes of the machine-language program and the source (C++) program. Proceed as follows:
 - a. Compile and link ("Build") the file to disk, creating on the disk an executable machine-language version, *hello.exe*.

b. In Windows Explorer, display your directory containing *hello.cpp*, and see also the *Debug* directory if it exists. File information is displayed showing the file name, size, file type, and date last modified.

c. What are the sizes of the following files?

(i) *hello.cpp* (ii) *hello.obj* (iii) *hello.exe*

2. A computer store buys books from a wholesaler for \$23 each. It must pay for each book received or return it. Write a C or C++ program that prompts for the number received and the number sold, and displays the amount owed and the number of books to be returned. Be sure to include a comment that describes the purpose of the program and submit a listing of the program and printed test results. Use meaningful identifiers for your variable names.

Sample I/O (user input is underlined):

```
Books received: 15
Books sold: 10
We owe the wholesaler $230
We must return 5 books.
```

3. Write a program that inputs five integers and displays their sum and their average.
4. The following C++ program contains ten syntax errors. Correct each one.

```
// mult.cpp
// multiplies two input values.
#include <iostream.h>
{
    cout << "Enter first factor: ";
    int input1; input2; product;
    cin << input1;
    cout << "Enter second factor: ";
    cin << input2;
    product := input1 * input2;
    cout << input1 << "*" << input2
        << = << product << endl;
};
```

Homework 4 (Numeric and character-based types; due 10/29)

Graded:

1. See project specification sheet. Write a properly documented program, with tests, that
 - declares variables of *appropriate types* (integer, string, or floating point) corresponding to the attributes of an inventory item;
 - prompts for and inputs these values from the keyboard,
 - computes value on hand,
 - displays these values in the format shown on the project spec sheet;
 - saves item attributes to a disk file, writing a space between each attribute (use file name that ends “.txt”, by adding “.txt” extension to file name after prompting user for file name without extension) – note that each value saved must be followed by white space to allow later retrieval;
 - retrieves attributes from disk file;
 - displays inventory item again, formatted.

Each of the steps under bullets 2-5 above should have a comment preceding it that describes the step. Include specification documentation, some of which can be from what you wrote for the graded part of Homework 2. For design documentation, describe the data items used and how they correspond to an actual inventory item.

Documentation & Code formatting	/ 20
Input	/ 20
Display	/ 20
File I/O	/ 20
Testing	/ 20

Ungraded:

2. Horstmann, p. 74, Ex. P2.6; p. 75, Ex. P2.14
3. Write a program that displays the hour it will be, *duration* hours after *start* o'clock, given those 2 inputs.

Sample I/O:

Start and duration? 2 23 It will then be 1 o'clock

4. Write a program that computes and displays the average speed and gas mileage for an automobile trip. Input distance traveled, time, and amount of gas, which may be fractional values. Display output, correct to one decimal place, in five labeled columns, three to echo input and two to present calculated values.

Sample I/O:

Distance: 200

Gallons: 10.5

Time: 4

Distance	Gallons	Time	Avg speed	MPG
200.0	10.5	4.0	50.0	19.0

5. Write a program that computes the radius of the circle whose area is input from the keyboard. Use the formula $Area = \pi \times radius^2$, where $\pi = 3.14159$ and the *sqrt* function defined in *math.h*. Display the radius correct to 2 decimal places.
6. Write a program that converts an input integer number of pounds plus an input integer number of ounces into a *double* number of kilograms. (1 lb. = 16 oz., one lb. = 0.454 kg). Display kilograms correct to 2 decimal places.
7. Write a program that prompts for a string and displays its first three characters and their numeric ASCII values. Submit listing, including a comment that describes the purpose of the program, and submit test results.
8. Write a program that prompts for two string variables, concatenates them together as the value of a third string variable, and displays both the resulting string and its length. You may choose the string library that you use: C-style (<*string.h*>) or ANSI/ISO C++ style (<*string*>).

Sample I/O:

X Windows

XWindows

Length: 9

9. Using appropriate data types, write one program that will do the following:
- (a) prompt for a floating point number;
 - (b) open a text file named *num.txt* for output, write

the number and close the file; and
(c) open *num.txt* for input, read the number, display the number, and close the file. Use the proper header files.

Homework 5 (Branch and loop statements; due 11/13)

Graded:

- Integrate the programs you wrote for the graded parts of Homeworks 3 and 4, so that your project is a single program that displays the main menu and responds to the user's menu choice. To do this,
 - Display the menu and input a value, as in the Homework-3 program;
 - Include an added menu choice, "Exit" or "Quit", that triggers loop termination when user chooses that option.
 - Write a multi-way branch statement that executes one of the operations from your Homework-4 program (input, display, save, retrieve), one *case* label per operation and using *named* constants for your case labels;
 - Validate each input if inventory data in some reasonable way (for example, don't accept certain negative values);
 - Put the menu-display, menu-input, and branch statements all into one loop;

Update your specifications documentation and, as part of your design documentation, describe the control structures you use and submit flowchart as in Homework 2, corrected, if necessary. Be sure to include comments in your *main*, which by now is a page long or more.

Documentation & Code formatting	/ 15
Menu loop	/ 30
Switch	/ 30
I/O	/ 10
Testing	/ 15

Ungraded:

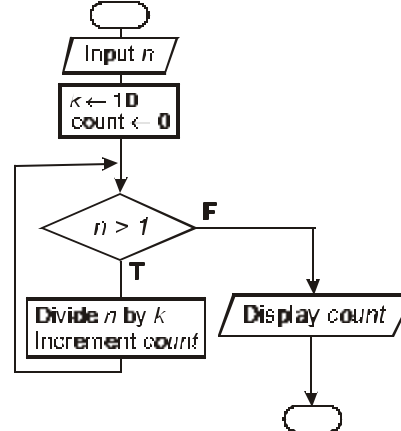
- Suppose an employer will pay tuition for a course on two conditions: (1) the tuition amount is not more than \$1000; (2) employee earns a 'B' or an 'A'. Write a program that prompts for tuition amount and final grade and tells whether the employee will be reimbursed. (For simplicity, suppose that the employee will enter 'A' for A or A-, and 'B' for B, B+ or B-.

- Write a program that prompts for a letter of the alphabet and uses a *switch* statement to convert it to the corresponding telephone dial digit, based on the following conversion table:

A B C	2
D E F	3
G H I	4
J K L	5
M N O	6
P R S	7
T U V	8
W X Y	9

Be sure to validate input.

- Write C or C++ code to implement the design in the flowchart below. Show test results for several values of *n* between 1 and 10,000.



- Write a program that prompts for integer *num* and displays a quantity of asterisks equal to *num*.
 Sample I/O:
How many stars? 4

- Write a program that loops to find the leftmost (most significant) digit of any input integer using type *int*.
 Sample I/O:
 Enter an integer: 215 2
 Enter an integer: 36 3
 Enter an integer: 7829 7
- Using a loop statement, write a program to compute and display the sum of six input real numbers.

Sample I/O:

DATA VALUE 1: 14.8
 DATA VALUE 2: -23.6
 DATA VALUE 3: 101.00
 DATA VALUE 4: 666.66
 DATA VALUE 5: -250.05
 DATA VALUE 6: -235.08
 SUM 743.87

8. Write a program that finds all integers a , b , and c greater than 0 in the range 1 to 50 such that $a^2 + b^2 = c^2$. Output should be a list of triples including (3, 4, 5) and (5, 12, 13), since $3^2 + 4^2 = 9 + 16 = 25$ and $5^2 + 12^2 = 25 + 144 = 169$.

9. The following program generates a random number in the range of 2 to 12, simulating the throw of two dice.

```
// dice.cpp
// Simulates roll of 2 dice and displays result.
#include <iostream.h>
#include <stdlib.h>
#include <time.h>

void main(void)
{
    /* Initialize random number generator's
       seed value, using current time: */
    time_t ltime;
    time(&ltime);
    srand(ltime);

    /* Display a value returned by
       random number generator: */
    cout << rand() % 6 + 1 + rand() % 6 + 1
         << " is a random number in range 2 to 12"
         << endl;
}
```

- (a) Assign the value of the random number to a variable and add a *switch* statement to have your program display “two” on a throw of 2, “three” on

3, etc.

(b) Modify program to initialize eleven variables: $n2..n12$, to the value 0, and after the throw of the dice to have your *switch* statement increment the variable that corresponds to the dice throw. For example, when a 7 comes up, add 1 to the variable $n7$.

(c) Start with your program for (b), which displays a random number from 2 to 12, simulating the throw of two dice. Modify it to generate 100 throws of the dice, storing in variable $n2$ the number of throws that result in 2, in $n3$ the number resulting in 3, etc., up to 12. Note: the switch statement you wrote for HW 6 should be *inside* your loop.

(d) Display the values of each of these variables labelled with the appropriate die toss.

Sample output:

Result	# tosses
2	5
3	7
4	6
.	.

(The values in the right-hand column will depend on random events.)

(e) Display a bar graph composed of eleven rows of stars (see problem 5), one row for each of the eleven counts. (Option: use an array)

Homework 6 (Subprograms; due 12/1)

Graded:

- (a) In your inventory application, move the code from each *case* label in your *switch* statement into a C++ function definition, replacing the code in the *switch* with a call to the function. Document the purpose of each function in a comment under or over the function header. Submit updated program documentation and test results as usual.

Function calls and definitions must have one parameter for each attribute of an inventory item. Functions that perform input must have reference parameters; others should have value parameters.

For the file-processing functions, you may choose to open your file in *main* and pass a stream object as a reference parameter, or open your file in the function and pass a file name as a parameter. Document these design decisions.

(b) The display function is to display the value on hand, computed from the quantity and price. Write a function that takes the appropriate parameters and returns this value to the display function.

(c) Submit your module hierarchy diagram from Homework 2, corrected if necessary.

Ungraded:

- Write a C or C++ function that accepts as parameters two values that may be fractional and passes their average back as a return value.
- Using a function call for each line, write a program that draws the triangle below on the screen. The function you write should accept a parameter that specifies the number of asterisks to display. Your program needs to define only one function other than *main*.
*
**

- Write and test a function that accepts a C-style string parameter and returns the number of *hyphens* in the string. You may declare the parameter either as *char** or *char[]*.
- Define and test a C++ function that prompts for the two (x,y) coordinates of a point on a plane and passes them both back as reference parameters in one call of the calling function.

Documentation & Pre-HW6 code	/ 15
4 function prototypes	/ 25
Calls	/ 25
Definitions	/ 25
Testing	/ 10

Homework 7 (Structure types and classes; ungraded)

1. Write a structure type declaration for circles.
A circle has an (x,y) location on the screen and a radius. Write and test a program that declares an instance of *circles*, assigns its x , y and *radius* members values input by the user, and displays the values.
2. Refer to the program below and modify the structure type *persons* so it is possible to store the weight of a person. Also, replace the output statements in *main* with a function call and write a function to display data about a person, by passing a *persons* item as a parameter. Test your changes. *Option*: Using the *class* keyword, write a C++ class to the above specifications.

```
// persn1.cpp
// Prompts for and displays person
// name and age.
#include <iostream.h>
struct persons
{
    char name[80];
    int age;
};

void input(persons& p)
{
    cout << "\nName: ";
    cin >> p.name;
    cout << "Age: ";
    cin >> p.age;
}

void main()
{
    persons student;
    input(student);
    cout << student.name << " is "
         << student.age << " years old.\n";
}
```

3. (*Challenge*) Below is a partial table of Red Sox batting performance as of April 10, 2002, showing name, at-bats, and hits. Create a text file of this data (e.g., copy-paste the web version, at www.framingham.edu/faculty/dkeil/cs1hwk.html). Write a program that opens this text file and reads each line into a structure or object, of a type that you can call *player*. For each player, compute batting average (at-bats / hits), displaying a table with the fourth (batting-average) column. Also compute batting average for this part of the team. (Suggestions: a player structure type will look very much like the *persons* structure type in problem 2. A good approach would be to write and call three functions that read a player line, compute average, and display player info. Each function can take a player object as a parameter or be a member of class *player*.)

Clark	17	5
Danon	21	7
Garciparra	23	8
Hillenbrand	20	7
Nixon	20	6
Ramirez	16	4
Sanchez	16	4
Varitek	15	6