Intro to Programming

Topic 9 - Part 2

Chapter 4

How to flowchart cin / cout

Examples:

```
cout << “Enter first age: ”;
cin >> ageOne;
```

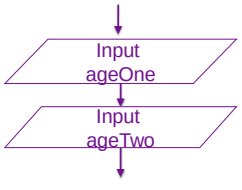
Note the spacing

```
cout << "Enter second age: ";
cin >> ageTwo;
```

Note the spacing

Example flow chart for cout/cin pair

- Each input on a flow chart will have a corresponding cout/cin pair
  - ➔ all other constructs are 1 - 1 (1 symbol per statement)



Note: we use the same variables as in the code!

endl vs. "\n"

Move the output to a new line

- They both accomplish essentially the same task
- endl → manipulator
- "\n" is an escape sequence

Program – Basic Structure

- ***Directive(s)*** **-** information the program needs (a list of all necessary header files used in the program)
- ***Heading*** **-** **int main ()**
  - *functions by definition return a value*

    → the above heading indicates that this function will return an int
- ***Inside the int main function*** **-**

```
{
    named constant declarations
    variable declarations
    executable statements
    return 0;
}
```


Identifiers → Review

- What are the two types of identifiers?
- Which can appear on the right side of a cout statement?
- Which can appear on the right side of a cin statements?

Declaring Identifiers → Review

- How do we declare variables?

Examples

- How do we declare constants?

Examples

Variables & Constants

- When is the amount of memory that will be allocated determined for a variable?
- When is the amount of memory that will be allocated determined for a constant?
- When is a value placed in a variable?
- When is a value place in a constant?
- Name two ways to assign values to a variable?

Examples

Data Tables

- What is a Data Table?
- What 2 things should the data table contain?

Example

Declaration Section Exercise

Write the necessary declaration section for a program requiring the following *variables and named constants. Use the proper style and be sure* to include the data table. Also, remember that identifiers must b descriptive.

- *a location to hold the name of the programmer (an unchanging value)*
- *a location to hold the date the code was written(an unchanging value)*
- *locations to hold the names of two users of the program (input from* the keyboard)
- *locations to hold the ages of each of the two users (input from the* keyboard)
- *a location to hold the older of the two (calculated & output)*
- *a location to hold the average of the ages (calculated & output)*

- *a location to hold the name of the programmer (an unchanging value)*
- *a location to hold the date the code was written (an unchanging value)*
- *locations to hold the names of two users of the program (input from* keyboard)
- *locations to hold the ages of each of the two users (input from the* keyboard)
- *a location to hold the older of the two (calculated & output)*
- *a location to hold the average of the ages (calculated & output)*

Assignment Statements - Review

Syntax

variableName = *expression*;

Assigns the expression to the variable.

Example:

```
ageOne = 15;
ageTwo = 23;
averageAge = (ageOne + ageTwo) / 2.0;
answer = 'y';
```

Note the spacing before and after all operators

Review

1. A compiler translates code written in a _____ language into _____ language.

2. Identifiers are associated with memory locations via _____.

3. A _____ is the name of a location in memory that has a data value that may be changed. Values for these identifiers are obtained at _____ time using an _____ statement or a _____ statement.

4. A _____ is the name of a location in memory that has a data value that may not be changed. These identifiers get their values at _____ time and may not appear on_____ or in a _____ statement.

Review

5. The documentation next to the declarations for variables and named constants is called the _____. It tells the reader _____ and _____ their values are obtained.

6. Each input block shown on a flowchart requires a _____ statement to prompt the user and a _____ statement to place the input value into the specified memory location.

7. Explain the difference between the following declarations

```
char charVal;
char strVal[10];
```