Students
Section 1 is a part of 2 assignments: CSC108 CH01 PA

Your work is saved. Your last submission was saved, but not graded (0/10). Resume this assignment before the due date to change it. Learn more.

Instructor
Points: 0/5
Due 01/05/2023, 11:59 PM EST

1.2 Programming basics

A first program

A simple C++ program appears below:

- **program** starts in main(), executing the statements within main's braces { } one at a time.
- **cout** (pronounced "see-out") prints to the screen. Text to be printed is in double quotes (") and ends with a semicolon and with a period.
- The **<<** operator indicates outputs are linked together (called **insertion**). The **endl** (pronounced "end-el") statement outputs a newline.
- The **return 0;** statement ends the program (the 0 tells the operating system the program ended without problems).

Code is the textual representation of a program, as seen below. Many code editors color words, as below, to assist humans to understand the meaning of the code.

The following code inserted into the top of a file enables the program to get input and output:

```
#include <iostream>
using namespace std;
```

INSTRUCTION 1.2.1: Program execution begins with main, then proceeds one statement at a time.

Start

Output

```
int main() {
   cout << "Hello,"
   cout << "World!"
   cout << endl;
   return 0;
}
```

Hello,
World!

Captions

1. A program begins executing statements in main(). Int wager 36;declares an integer variable wager. int wager = 36; assigns wager with 36.
2. The cout statement outputs the value of wager, 36, to the screen. Cout << wager; outputs wager's value.
3. The cout statement outputs the value of wager, 36, to the screen. Cout << "\n"; outputs a newline.
4. The return 0; statement ends the program.

INSTRUCTION 1.2.2: A first program.

Consider the program above.

1) Program execution begins at main(), and executes statements

Hint: Only one statement is executed at a time.

- main
- int
- cout
- endl

2) The statement int wager; creates a variable named wager that is

Hint: Consider the program above.

- wager
- int
- cout
- endl

3) Assume the following order of statements work the same as above?

Hint: Order matters.

- endl
- cout
- int

4) Cout << wager; outputs

- the variable
- the value
- nothing

5) The statement wager = 36;"-1]] resulted in what value?

- 36
- 37
- 36"-1]]

6) A C++ statement outputs literals to ..... .

- a file, cannot output text
- the keyboard
- the screen

Feedback?

zyDE 1.2.1: A first program

Below is the zyBooks Development Environment (zyDE), a web-based programming practice environment. Click run to compile and execute the program. Next observe the output. Change 20 to a different number (say 30 and click run again) to see the different output.

Load default template. Run

```
#include <iostream>
using namespace std;

int main() {
   int wager = 36;
   cout << "Hello,";
   cout << wager << " is a";
   cout << " good";
   cout << " number." << endl;
   return 0;
}
```

Feedback?

Basic input

Programs commonly get input values, perform some processing on that input, and put output values to a screen or elsewhere. Input is commonly gotten from a keyboard, a file, fields on a web form or app, etc.

The following statement gets an integer value and puts that value into variable **x**. **cin** is short for characters in.

INSTRUCTION 1.2.3: A program can get an input value from the keyboard.

Start

Output

```
int main() {
   int x;
   cin >> x;
   cout << "Hello,";
   cout << x << " is a";
   cout << " good";
   cout << " number." << endl;
   return 0;
}
```

Hello,

Captions

1. The cin >> wager statement gets an input value from the keyboard (or file, etc.) and puts that value into the wager variable.
2. wager's value can then be used in subsequent processing and outputs.

Feedback?

INSTRUCTION 1.2.4: Basic input.

1) Which statement gets an input value into variable numCats?

- int numCats;
- cin >> numCats;
- int >> numCats;

Feedback?

INSTRUCTION 1.2.5: Basic input.

1) Type a statement that gets an input value into variable numHikers.

Assume numHikers has been declared.

Check Show answer

Feedback?

zyDE 1.2.2: Basic input

Find the program and observe the output. Change the input value from 3 to another number, and then run again. Note: Handling input from a human is very error-prone, and thus programs should check for valid input. Checking the input is a professional's job. For students, assume input characters use a bug-free keyboard and environment.

Load default template. Run

```
#include <iostream>
using namespace std;

int main() {
   int x;
   int numHikers;

   cin >> x;
   cin >> numHikers;

   cout << "Hello,";
   cout << x << " and";
   cout << " about";
   cout << " " << numHikers << " people";
   cout << " need shirts";
   cout << " and pants.";
   cout << endl;
   return 0;
}
```

Feedback?

Basic output: Text

The **cout** operator's output, cout is short for characters out. Outputting text is achieved via cout << "desired text"; . Text in double quotes " " is known as **string literal**. Multiple cout statements continue printing on the same output line. The statement cout << endl; prints a newline. Escape sequences **'\n'** and **'\t'** are used for "newline" and "tab" escape sequences, respectively. Escape sequences begin with a backslash ('\') followed by a letter. Escape sequences are used to type special characters.

Figure 1.2.1: Outputting text and newlines.

```
#include <iostream>
using namespace std;

int main() {
   cout << "Hello,";
   cout << "World!";
   cout << endl;
   cout << "Hello,";
   cout << "World!";
   cout << endl;
   return 0;
}
```

Output

```
Hello,World!
Hello,World!
```

```
#include <iostream>
using namespace std;

int main() {
   cout << "Hello,";
   cout << "World!";
   cout << endl;
   cout << "Hello,";
   cout << "World!";
   cout << endl;
   return 0;
}
```

Output

```
Hello,World!
Hello,World!
```

The notation cout << ... gives the appearance of the item on the right being "streamed" to cout like items flowing along a stream into a lake, where cout represents the computer's screen.

INSTRUCTION 1.2.6: How to use cout (spaces and line).

Start

Output

```
int main() {
   cout << "Hello,";
   cout << "World!";
   cout << endl;
   cout << "Hello,";
   cout << "World!";
   cout << endl;
   return 0;
}
```

Output

```
Hello,World!
Hello,World!
```

Captions

1. endl (spelled with a lowercase "L" at the end) starts a new output line, meaning the cursor and following output move to the next line.
2. The endl statements create a blank line in the output.

Feedback?

INSTRUCTION 1.2.7: Basic text output.

1) What statement outputs Welcome!

- cout << Welcome;
- cout << "Welcome";
- cout << "Welcome!";

2) What statement starts a new output line?

- cout << endl;
- cout << "\n";
- cout << endl;

INSTRUCTION 1.2.8: Basic text output.

End each statement with a semicolon. Do not output a new line unless instructed.

1) Type a statement that outputs Hello.

Check Show answer

2) Type a statement that starts a new output line.

Check Show answer

Feedback?

Outputting a variable's value

Outputting a variable's value is achieved via cout << x; . Note that no quotes surround x.

Figure 1.2.2: Outputting a variable's value.

```
#include <iostream>
using namespace std;

int main() {
   int x = 0;
   int y = 0;

   cout << "Hello,";
   cout << x << " and";
   cout << " " << y << " people";
   cout << " need shirts";
   cout << " and pants.";
   cout << endl;
   return 0;
}
```

Output

```
Hello,0 and 0 people need shirts and pants.
```

Feedback?

Note that the programmer intentionally did not start a new output line after outputting "Hello,". Is that the wage variable's value would appear on that same line.

INSTRUCTION 1.2.9: Basic variable output.

1) Given variable numCats = 5, which statement outputs 5?

- cout << "numCats";
- cout << numCats;
- cout << numCats;

Feedback?

INSTRUCTION 1.2.10: Basic variable output.

1) Type a statement that outputs the value of numHikers (a variable).

End statement with a semicolon. Do not output a new line.

Check Show answer

Feedback?

Outputting multiple items with one statement

Programmers commonly use a single output statement for each line of output by combining the outputting of text, variable values, and a new line. The programmer simply separates the items with << symbols. Such combining can improve program readability because the programmer code corresponds more closely to the program's output.

Figure 1.2.3: Outputting multiple items using one output statement.

```
#include <iostream>
using namespace std;

int main() {
   int x = 0;
   int y = 0;

   cout << "Hello,"
        << x << " and"
        << " " << y << " people"
        << " need shirts"
        << " and pants."
        << endl;
   return 0;
}
```

Output

```
Hello,0 and 0 people need shirts and pants.
```

Feedback?

zyDE 1.2.3: Single output statement

Modify the program to use only one output statement, one for each output sentence.

```
to 2014, the driving age in LA,
10 states have exceptions.
```

Do not type numbers directly in the output statements; use the variables. WHICH, Make incremental changes: Change one code line, run and check, change another code line, run and check, repeat. Hint: It is a change everything at once.

Load default template. Run

```
#include <iostream>
using namespace std;

int main() {
   int numYear = 2014;
   int numStates = 10;
   int numExceptions = 10;

   cout << "to " << numYear << ",";
   cout << " the driving age in LA,";
   cout << " " << numStates << " states";
   cout << " have exceptions.";
   cout << endl;
   return 0;
}
```

Feedback?

INSTRUCTION 1.2.11: Basic output.

Indicate the actual output of each statement. Assume variable is 23.

1) cout << "It's now " << variable << " years.";

- It's now 23 years.
- It's now variable years.
- Syntax error.

2) cout << variable << "years to pass.";

- 23years to pass.
- 23 years to pass.
- Syntax error.

INSTRUCTION 1.2.12: Output simulator.

The tool below supports a subset of C++, allowing for experimenting with cout statements. The activity is marked as complete upon successfully with the tool.

The following variable has already been declared and assigned: unsolicitedMessageCnt = 1000000; . Using that variable (do not type the long number) along with text, fix the cout statement to output the following:

```
Only 1,000,000 ads remaining.
```

There is no semicolon, this

```
CONGRATS!!! YOU WON!!!
```

```
cout << "Change this string!"
```

Change this string!

Feedback?

INSTRUCTION 1.2.3: Enter the output.

Start

Type the program's output

```
int main() {
   cout << "Run is happy.";
   cout << endl;
   return 0;
}
```

Run is happy.

Clear Reset

INSTRUCTION 1.2.2: Read multiple user inputs. Full screen

Using input and output

The **cin** operator reads the **input** elements in the corner order to read input values into symbolicals and then continues. Then, cout and cout.flush() **output** the elements to output the results, cout.flush() and the user. End with a newline by outputting endl.

So, if the input is 2 2001, then the output is

```
2 2001
```

Note: Can autopilot automatically runs your program several times, trying different input values each time to ensure your program works for any values. Thus, the program is tested twice. Type the input 1 and 2000, and then with the input 2 and 1950.

How to use this tool

Unused

```
int x = 0;
int y = 0;
int z = 0;
int numHikers;
int numCats;
int numDogs;
int numFish;
int numHamsters;
int numLizards;
```

main.cpp Load default template

```
#include <iostream>
using namespace std;

int main() {
   int x = 0;
   int y = 0;
   int numHikers;

   cin >> x;
   cin >> numHikers;

   cout << "Hello,";
   cout << x << " and";
   cout << " about";
   cout << " " << numHikers << " people";
   cout << " need shirts";
   cout << " and pants.";
   cout << endl;
   return 0;
}
```

Check

INSTRUCTION 1.2.13: Programming basics

Begin

This challenge activity consists of a series of auto-gradeable, randomized questions allowing unlimited submissions. You must correctly answer a question at least twice before proceeding to the next level. The purpose of this first level is to familiarize students with the zyBooks programmer environment.

Example code to the first question:

Step 1

```
int x = 0;
int y = 0;
int z = 0;
int numHikers;
int numCats;
int numDogs;
int numFish;
int numHamsters;
int numLizards;
```

Step 2

```
int x = 0;
int y = 0;
int z = 0;
int numHikers;
int numCats;
int numDogs;
int numFish;
int numHamsters;
int numLizards;
```

Step 3

```
int x = 0;
int y = 0;
int z = 0;
int numHikers;
int numCats;
int numDogs;
int numFish;
int numHamsters;
int numLizards;
```

main.cpp (read-only) AUTOSAVED (0/10)

```
#include <iostream>
using namespace std;

int main() {
   int x = 0;
   int y = 0;

   // Your code goes here \/

   return 0;
}
```

1 2 3 4 5

Check Next level

Feedback?

Newline character

A newline is a character produced by inserting a backslash and **newline character** within a string literal. Or, Outputting "\n\n\n" produces each number on its own output line. Or produces of other characters. Letters, the together are considered as one newline character. There are technical advantages to using either and or if to code newlines. Can determine when each approach is good

How was this section? Provide section feedback

Activity summary for assignment: CSC108 CH01 PA

Your work is saved. Your last submission was saved, but not graded (0/10). Resume this assignment before the due date to change it. Learn more.

This assignment's due date has passed. Activity will still be recorded, but will not count towards this assignment (unless the due date is changed). See details for more info.

Completion details

106 / 106 points
106 / 106 points (ignoring optional)
No deadline set