SIT102 Introduction to Programming

Pass Task 2.2: Hand Execution

Overview

Using the assignment statement, you can assign a value to a variable. In this task you will demonstrate how this action works within the computer. Hand execution is the process of stepping through a block of code to demonstrate how it runs within the computer. Being able to read through code and understand what it does is a really important skill to develop. It will help you locate issues, but also helps when you are designing and writing code as you understand how it will operate when it is executed.

Submission Details

Use the instructions on the following pages to step through the supplied code, and document how the values stored in the variables change over time.

Submit the following files to OnTrack.

- Images of your hand execution process

The focus of this task is on the declaration of local variables, and use of the assignment statement to store a value within the variables.

Instructions

Watch the 2.2P Hand Execution and follow the steps there. The following steps outline the details shown on the video, so either follow the steps in the video or use the following as a guide. You can skip to the end for the additional instructions if you have followed the steps in the video.

Demonstrate how the following snippets of code execute when they are run by the computer.

1. Get a piece of paper, and a pencil (or pen).
2. Use the Hand Execution process (see Hand Execution video above) to demonstrate how the following programs work. The basic process is:
    - Step through the code one statement at a time - like the computer does.
    - Carry out each statement, based on what the statement does:
        - **Variable declaration**: Allocates memory for the variable. Do this by drawing a box to represent the memory allocated to each variable. Write the name of the variable next to the box so you can keep track of which “variable” is which. This variable now represents this piece of memory, or space on the page in our case.

          Keep the box size fairly large, as we will use it to keep track of the values written to the variable as the program runs. When the value changes you will need to cross out the old value and write in the new value.
        - **Assignment statement**: Sets the value of the variable, changing the value stored at its location in memory. In your case, you assign a value by crossing out the old value and writing the new value in the variable’s box on the page (its memory). Remember, don’t draw a new box, just cross out the old value in the box and then write in the new value. If you run out of space in the box you can write the value next to it, as long as we can tell which value belongs to which variable.
        - When the variable is used within an **expression**, read the value from its box (its memory) and use that. Remember the value may change, but you always want to just read the value at the point in time when the variable was used.
3. When you finish hand executing each program, take a photo (or scan) the page. You will upload each of these as an image to OnTrack.

For each program you are aiming to show the *process* the computer followed as well as the final state. You will be able to see that process by reviewing how the values in the variables were updated (by reviewing the values that were crossed out).

Program 1

```
int main()
{
    int a, b, c;

    a = 3;
    b = 9;
    c = a + b;
    b = 20;

    write("c is ");
    write_line(c);

    return 0;
}
```

Hand execute this code and grab a picture of the result to upload to OnTrack.

1. Start at the beginning of main.
2. The first line declares 3 variables, so create three large-ish boxes on the page. Label them a, b, and c. These spaces now represent the three different variables created in the program.
3. The next line assigns the value 3 to a, so write 3 in the “a” box.
4. Similar actions occur on the next line.
5. When `c` is assigned, the expression reads the values of `a` and `b`. Look in the boxes on your page to find the values… then add them together and store the result in `c`.
6. The next line then assigns a new value to `a`. Cross out the old value from the `a` box and write in the new value.
7. Indicate what is output by the write_line procedure call.

What you have done here mirrors what the computer would do when it runs this code. Grab a picture or scan of your paper with the hand execution of this program to submit to OnTrack.

Program 2

Follow the same hand execution process to execute the following:

```
int main()
{
    int x;
    x = 8;

    write(x - 1);
    write(" ");
    write_line(x);

    return 0;
}
```

Remember, the only way a variable changes is if you have an assignment statement.

Program 3

Hand execute the following code, keeping the idea of **sequence** in mind. Make sure to indicate what the output of the `write_line` calls are.

```
int main()
{
    int salmon, dumplings, meals;

    salmon = 0;
    dumplings = 0;
    meals = salmon + dumplings;
    salmon = 30;
    dumplings = 70;

    write("Fish: ");
    write(salmon);
    write("Dumplings: ");
    write(dumplings);
    write("Meals: ");
    write_line(meals);

    return 0;
}
```

Program 4

Hand execute this program code. Remember it just runs in sequence.

```
int main()
{
    int a, b, c, d, e;

    a = 4;
    b = 8;
    c = 12;
    d = a;
    e = 30;

    b = a;
    a = a + b;
    c = e - c;
    d = a;
    e = e - 1;

    return 0;
}
```

Task Discussion

Discuss the following with your tutor to demonstrate your understanding of the concepts covered.

- Describe how sequence has an impact on the output of these different programs.

- Ask any questions about the process, you will need to demonstrate this in the tests.