GERMANNA
ACADEMIC CENTER
FOR EXCELLENCE

Computer Programming Problem Solving Process

Creativity and problem solving play a critical role in computer programming. It is important to apply a structured process to identify problems and generate creative solutions before a program can be developed. This handout will explain the process of approaching a problem and developing a solution for it.

How to begin the problem-solving process

Programmers must solve many different types of problems. One problem that students often encounter in programming classes is how to find the lowest or highest value in a list of numbers. Review the following example, and note the steps a programmer could take to solve the problem.

Example problem:

Find and display the largest number in this list:

```
[ 1218, 9957, 1949, 5683, 2665, 5382, 1322, 9458, 4061, 8400,
6955, 1419, 8585, 211, 5299, 5600, 1414, 285, 9515, 5989, 3387,
7613, 3489, 8499, 9981, 1448, 8976, 2363, 564, 9481, 784, 800,
1018, 6613, 8448, 7591, 9561, 983, 1605, 8979, 4068, 4978, 4050,
3479, 5356, 8644, 3235, 438, 8417, 432, 4478, 1619, 2075, 66,
1912, 6749, 127, 9492, 5964, 9854, 646, 3730, 2335, 2105, 6566,
3898, 7832, 8238, 5361, 4237, 7791, 1261, 7227, 8850, 9801, 9511,
9109, 5453, 8071, 65, 4590, 8377, 6805, 1060, 5293, 8673, 4219,
440, 6102, 9931, 2185, 2745, 6261, 9513, 2981, 491, 5477, 1426,
7276, 664, 5996, 6232, 5380, 4085, 1685, 9611, 6102, 431, 505,
8748, 5631, 2704, 8309, 702, 3741, 4178, 6896, 326, 932, 3854,
4567, 870, 3721, 1460, 2563, 8607, 5230, 2947, 5171, 6248, 1383,
3637, 1952, 1513, 2170, 2852, 3450, 3282, 3607, 8896 ]
```

Step 1 – Identify the problem that must be solved.

The first step is to identify the problem that needs to be solved. In this example, the largest number in the list must be found and displayed.

Step 2 – Understand what the problem presents.

The problem presents a list of numbers. A quick glance at the list shows all the numbers to be positive. Thus, it is a list of multiple positive numbers.

Step 3 – Determine how to solve the problem.

There are multiple approaches that could be taken to solve any given problem. In this example, the number with the largest value must be found, so one solution would be to cycle through each number and determine if it is the largest value. This can be accomplished by comparing the numbers and discarding the ones that have lower values.

Step 4 – Develop a method to achieve the solution.

It is established that all the numbers are positive, meaning that none of the numbers will be less than the value of 0. A process can be used that will compare each number in order of occurrence to a check number that will initially be set to be 0. If the number being compared has a higher value than the check number, then that value will be the new check number. Then, proceed to repeat the process with each subsequent number in the list. When the process is finished, the number with the highest value will be the final check number.

The solution can be obtained through the following process.

> **First, make a check number; set it to 0.**
>
> The check number needs to be lower than the highest value, but the highest value is currently unknown. In this example, there are no negative signs (-) on any of the numbers; thus, there are no negative numbers. Therefore, it can be assumed that 0 will be lower than the highest number in the list.
>
> **Second, look at each number in the list and compare it to the check number.**
>
> If a number is greater than the check number, change the check number to be the same as the number being examined; however, if a number is less than the current check number, leave the check number unchanged.
>
> The following sequence shows how those steps would be applied to the list:
>
> 1. Check number = 0.
> 2. The first number in the list: 1218.
> 3. 1218 is greater than 0, so the check number is changed to 1218.
> 4. The check number is now 1218.
> 5. Move to the next number in the list.

```
6.    Next number in the list: 9957.
7.    9957 is greater than 1218, so the check number is changed to 9957.
8.    The check number is now 9957.
9.    Move to the next number in the list.
10.   Next number in the list: 1946.
11.   1946 is not greater than 9957, so the check number is left unchanged.
12.   The check number remains 9957.
13.   Move to the next number in the list.
```

Continue the process until each number in the list has been examined.

Lastly, display the final check number.

When the process is complete, the check number will be the largest number. For this example, it is 9981. Therefore, 9981 will be displayed.

How to use pseudocode during the planning process

Pseudocode informally describes the step-by-step process that will be implemented to solve a problem or accomplish a task, and it is an integral part of the planning stage. The pseudocode for this example is as follows:

```
Make a check number; set it to 0.
Look at each number in the list and compare it to the check number.
      If a number is higher than the check number, change the check
      number to that number. If it is not, the check number remains the
      same.
            Repeat this step until each number in the list has been
            checked.
Display the final check number.
```

For more information about how to write pseudocode, please refer to the Academic Center for Excellence's Pseudocode and Flowcharts handout.

The blocks of code shown below demonstrate how the steps would be written in the Python, Java, and C++ programming languages. The "listOfNumbers" variable that is featured in the code represents the list of numbers that were presented in the problem.

Python Example:

```
# Make a check number; set it to 0
checkNumber = 0

# Loop through the list of numbers and compare every number in the list to
# the check number
for number in listOfNumbers:

    # If the number being compared to the check number is larger than the
    # check number
    if number > checkNumber:

        # The number being compared is the new check number
        checkNumber = number

# Display the final check number
print(checkNumber)
```

Java Example:

```
// Make a check number; set it to 0
int checkNumber = 0;

// Loop through the list of numbers and compare every number in the list to the
// check number
for (int i = 0; i < listOfNumbers.length; i += 1) {

      // If the number being compared to the check number is larger than the
      // check number
      if (listOfNumbers[i] > checkNumber) {

            // The number being compared is the new check number
            checkNumber = listOfNumbers[i];

      }

}

// Display the final check number
System.out.print(checkNumber);
```

C++ Example:

```
// Make a check number; set it to 0
int checkNumber = 0;

// Determine how many numbers are in the list
int getListLength = sizeof(listOfNumbers)/sizeof(int);

// Loop through the list of numbers and compare every number in the list to the
// check number
for (int i = 0; i < getListLength; i++) {

	// If the number being compared to the check number is larger than the
	// check number
	if (listOfNumbers[i] > checkNumber) {

		// The number being compared is the new check number
		checkNumber = listOfNumbers[i];

	}

}

// Display the final check number
cout << checkNumber;
```

The Academic Center for Excellence (ACE) offers free on-campus or online tutoring appointments. For further assistance with programming concepts, please call ACE at (540) 891-3017 or email us at ACE@germanna.edu.