Visual Pro Tutorial B

A beginners guide to Visual Scripting

VisualPro: A Lightweight; Visual Scripting Tool

Tutorial: Functions and Variables

Authors:
Edward Patch
Student Number: 1801492

Supervisor:
Mike Dacey

16 February 2022

Contents

1 Learning Objectives

The following learning objectives are as follows:

1. To declare functions in a Functional Programming (FP) style rather than an OOP style.
2. To declare variables in both global scope **(considered bad practice)** and function scope.

2 Introduction

Focus on the FP style, which gives the chance to increase existing skill level to explore both OOP and FP style languages. Supported languages in VisualPro are C and C++.

Tip 2.1: Fun Fact:

C++ is an extension of C that offers both FP and OOP style language, making the language a powerful language.

2.1 What is the Differences?

OOP style enables the creation of objects, whereas FP style contains functional structure. However, C does allow the creation of a 'struct' that is similar to a class, however, it only allows variables. It is great for creating reusable data structures but do not Constructors or Deconstructors that play a big part in memory management.

An example of C# (OOP Style) vs. C (FP Style):

Example 2.1: OOP Style

```
using System; // Provides the Console.WriteLine function from
    the 'System' library.

class Program {
  static void Main(string[] args) {
    int x = 6, y = 45;
    Console.WriteLine("x + y = " + (x * y).ToString());
    // result: 270 (The ToString method converts the x * y
        calculation to a string and appends to the string).
  }
}
```

Example 2.2: FP Style

```
#include <stdio.h> /* Provides the printf function from the
'stdio' library. */

int main(int argc, char** argv) {
    int x = 6, y = 45;
    printf("x + y = %i", x * y); // result: 270 (%i means first
        parameter is an integer and includes it to the string).

    return 0;
}
```

Note that the OOP style relies on encapsulation, whereas the C language or any other language is FP.

3 VisualPro Environment

3.1 Features

VisualPro offers a few features such as:

- Classes, Functions and Arguments, and Variables.
- Saving in Multiple Languages.
- Drag and Drop Elements and Text Areas.
- Property Windows to Control Arguments and Relationships.

3.2 Known Bugs

A couple of bugs include:-

- Arguments for functions are not currently available.
- Deleting containers does not mathematically reset the following location or move existing containers backwards.

3.3 Saving Progress

After completing a tutorial, select the language and press save.

Tip 3.1: Compatibility

As mentioned previously, not all languages support object-orientation. If a language is not compatible with a particular keyword, the code generator will ignore the selected syntax object.
Example Language: C is not supported.

4 Terminology

4.1 What is a function?

Like a method, a function exists in the global scope of an FP style language, such as the C language. A function can return data like an integer or a string or a 'void' data type that returns nothing. It is also important to note that arguments are the function's inputs that enable data to pass through the function scope.

Three examples of different functions and their purposes:

Example 4.1: Function: VOID Return Type

```
#include <stdio.h>

// The function returns nothing.
// The function accepts no arguments.
void PrintHelloWorld()
{ printf("Hello World"); }

int main(int argc, char** argv)
{
  // The call to the function.
  PrintHelloWorld();
  return 0;
}
```

Example 4.2: Function: VOID Return Type — Integer Argument

```
#include <stdio.h>

// The function returns nothing.
// The function accepts one argument (integer).
void PrintX4(int x)
{ printf("%i * 4 = %i", x, x*4); }

int main(int argc, char** argv)
{
  PrintX4(10); // Result: 10 * 4 = 40
  return 0;
}
```

Example 4.3: Function: Integer Return Type — Integer Argument

```
#include <stdio.h>

// The function returns an integer.
// The function accepts two arguments (integer and integer).
int PrintXY(int x, int y)
{ return x * y; }

int main(int argc, char** argv)
{
  int x = PrintXY(2, 7);
  printf("2 * 7 = %i", x); // result: 2 * 7 = 14

  return 0;
}
```

5 Functional Structure

5.1 Exercise: Understanding the Basics

A function exists outside of a class and within the global scope. When declaring and defining a function, the location of the declaration must be above the definition of the function.

A couple of examples are on display below:

Example 5.1: Declaration of a Function (A):

```
#include <stdio.h>

// Declaration and Definition
float XMultiplyY(double x, double y) {
    return x * y;
}

int main(int argc, char** argv) {
  printf("8.87 * 2.65 = %f", XMultiplyY(8.87, 2.65));
  // result: 23.505501
  return 0;
}
```

Alternatively, defining a function is to declare above the 'main' function and defining the function below the 'main' function. For example:

Example 5.2: Declaration of a Function (B):

```
#include <stdio.h>

// Declaration
float XMultiplyY(double x, double y);

int main(int argc, char** argv) {
  printf("8.87 * 2.65 = %f", XMultiplyY(8.87, 2.65));
  // result: 23.505501
  return 0;
}

// Definition
float XMultiplyY(double x, double y) {
    return x * y;
}
```

Also, the declaration could belong to a header file, including the definition with the header tag.

Tip 5.1: Evidencing

Necessary: Make sure to copy each exercise code to the survey when asked. A reference of the survey question helps find the corresponding survey question to paste the final code.

5.2 Exercise: Toaster Functionality

A toaster has three primary functions, heat up, cool down and timer. First, let us drag a function onto the Work Area panel within VisualPro. Name this function, 'HeatUp'. A functional structure does not have Access Modifiers, so leave this field blank. Now, set the return value (datatype) as void. Drag another function onto the Work Area panel, naming the function 'CoolDown' and set the return value as void. Finally, drag the last function onto the Work Area panel, naming the function 'Timer' and set the return value as bool (for Boolean (true/false)).

Make sure to save the file in the C language and upload the file's contents to the relevant section of the survey.

Did it look like this visually?

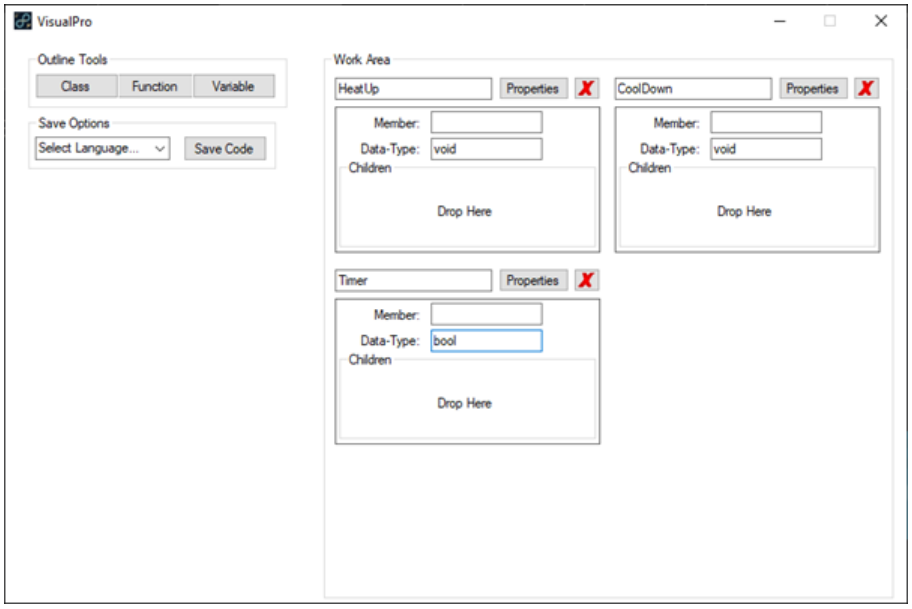

Figure 1: VisualPro - Exercise A

Expected Code Generation:-

This code should look similar to this:

Example 5.3: Toaster Functionality

```
void HeatUp()
{

}

void CoolDown()
{

}

bool Timer()
{

}
```

5.3 Excercise: Toaster Variables

When thinking about a toaster, think of a few variables. There is no right or wrong answer to this question. Create the variables on a blank VisualPro pad, by dragging and dropping the variables on the Work Area panel. Again, there is no need for a member.

Note: This way, the variables are within the global scope and is not good practice. **Make sure to save the file in the C language and upload the file's contents to the relevant section of the survey.**

Did it look like this visually?

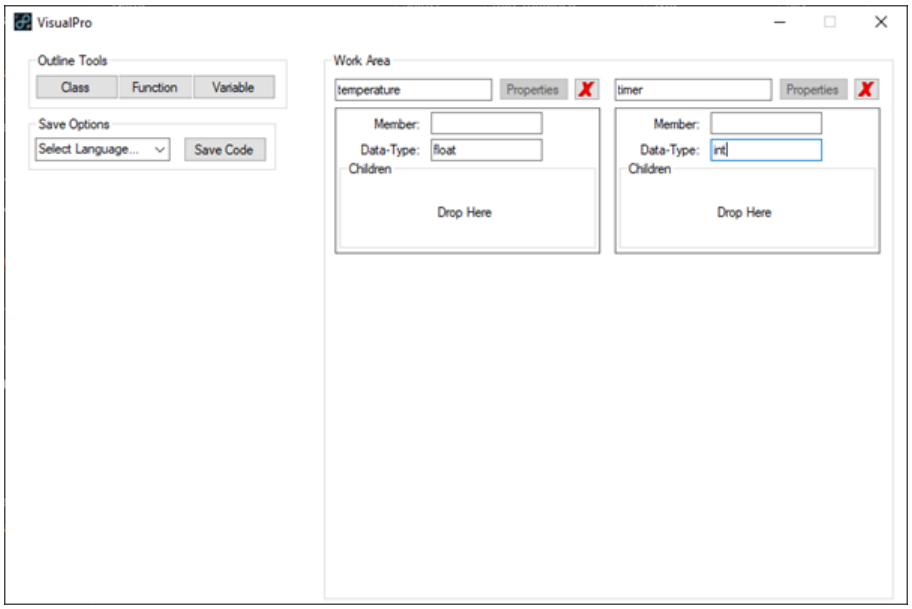

Figure 2: VisualPro - Exercise B

Expected Code Generation:-

Example 5.4: Toaster Variables

```
float temperature;
int timer;
```

5.4 Excercise: Fuel Station

Scenario: A fuel station has many factors; try creating functions with function scoped variables that mimic a fuel station. Try dragging variables to the Function's Work Area panel. Again, there are no right or wrong answers.

Make sure to save the file in the C language and upload the file's contents to the relevant section of the survey.

Did it look like this visually?

Expected Code Generation:-

Example 5.5: Fuel Station

```
double Cashier() {
  int availablePumps;

  return 0.00;
}

int GetAvailablePumps() {

}

double GetFuelLevel() {

}
```

Tip 5.2: Hint: Function and Variable

A potential function requires a cashier that returns a double return type. (A cashier would exchange change if any is required.) A variable within this function could contain availablePumps as a integer variable.

5.5 Exercise: Try It Yourself (TIY)

Great! To continue this exercise, think of a scenario and drag and drop functions and variables. **Make sure to save the file in the C language and upload the file's contents to the relevant section of the survey. Please state the objective within the survey.**

6 Keywords

- FP - Functional Programming.
- OOP - Object-Oriented Programming.

References

[1] cplusplus.com, "Classes - C++ Tutorials." [Online]. Available: https://www.cplusplus.com/doc/tutorial/classes/

[2] Bill Wagner, "Classes," Sep. 2021. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/types/classes

[3] Word Disk, "List of object-oriented programming languages - Wikipedia @ WordDisk," Mar. 2018. [Online]. Available: https://worddisk.com/wiki/List_of_object-oriented_programming_languages/