USING SUBPROGRAMS

Hieu D. Vu, Ph. D.
Hubei University of Economics
School of Information Management
No 8 Yangqiaohu Rd., Jiangxia
Wuhan, PRC
hieu.vu52@yahoo.com

Abstract

The central idea in computer programming is problem solving. There are several techniques, but the most fundamental technique in problem solving is to break down a big problem into pieces of smaller problems that can be solved easily. These small pieces of problems called subprograms, a general term in any programming languages.

Keywords:

Programming, Subprogram(s), Procedure(s), Function(s), Method(s)

I. INTRODUCTION

The foundation concept of computer programming is to break down a program into smaller subprograms and each subprogram only handles a single task. A major advantage of writing subprograms is to avoid repetition, we can use (call) a subprogram several times instead of duplicating of codes. Depending on the language, these subprograms maybe called: subroutines, remote blocks (Fortran), or functions, procedures (Pascal). The "C family" that includes: C, C++, and C# call functions. Java calls methods. This research paper examines how computer programs were constructed with subprograms in the "old way" (Pascal) and the "new way" as in C, C++, C#, and Java.

II. BACKGROUND

Pascal was developed by a Swiss computer scientist, Niklaus Wirth in the early 1970s, and the language was named after the French mathematician Blaise Pascal. A standard version of Pascal was approved in 1983 by American National Standards Institute (ANSI) and Institute of Electrical Electronic Engineers (IEEE). Pascal was a small compiler and with the growing of microcomputer (PC), Pacal became popular, the principle language for teaching programming until late 1990s.

In the early day, a Pascal program must conform to the program's layout as:

```
LABEL section
CONST section
TYPE section
VAR section              (declare global variables)
PROCEDURE(s) and FUNCTION(s) section
```

The sections above is called declaration part, then the main body of the program that starts with the keyword BEGIN follows. In this format layout, a procedures or functions must be defined before the callings, and the main body of the program which contains principal logics was placed at the bottom.

Example:

```
PROGRAM SimpleProc (OUTPUT);
{---Beginning Procedure---}
PROCEDURE DrawLine;
CONST
Dash = '-';
LineLength = 20;
VAR
Counter :INTEGER;
BEGIN
FOR Counter := 1 TO LineLength DO
      WRITE(Dash);
WRITELN
END;
{---End of Procedure---}
{--- Main program---}
BEGIN
WRITELN;
DrawLine;
WRITELN('** THIS IS A TEST **');
Drawline
END.
```

The output:

```
--------------------
** THIS IS A TEST **
--------------------
```

[1]

This is somewhat inconvenience and awkward to program, the programmer after coding the calling statement, must go back to the declaration section to write the subprogram and the entire program appears “up-side-down”.

A good program should be lay outin logical order, easy for reading, writing and control such as we read a paper or a book from top to bottom. Another aspect in good programming technique is to transfer the controls downward not upward.

III. USING SUBPROGRAMS IN NEWER PROGRAMMING LAGUAGES

From the example on previous section, we can conclude that not only variables, also procedures and functions to be used in the program, they must be declared before can be used. The following sections, we will exam in to see how newer programming languages deal with this “up-side-down” problem.

Another point that needs to mention here before we continue.In newer programming languages such as C, C++, C#, there is no-procedure, everything is called function. A function can return at most one value or none (void function which is equivalence to a procedure in Pascal). Java calls a subprogram “method” instead of function, but it is still the same.

III. 1. C-LANGUAGE, C++, AND C#

1. In C, a function declaration informs the compiler that a function is being used in the program and the body of the function is defined separately. The declaration of a function looks exactly the function header but it ends with a semicolon and no implementation (body of the function).

```
      Return Type function Name(parameterList);//Declaration format
example:
      int max(int n1, int n2);
or:
      int max(int, int);     //Names are not necessary
```

Example full program:

```
      #include <stdio.h>

      /* function declaration */
      int max(int num1, int num2);

      int main (){
      /* local variable definition */
      int a = 100;
      int b = 200;
      int ret;
      /* calling a function to get max value */
      ret = max(a, b);
      printf( "Max value is : %d\n", ret );
      return 0;
}
```

```
/* function returning the max between two numbers */
int max(int num1, int num2){
/* local variable declaration */
int result;
if (num1 > num2)
result = num1;
else
result = num2;
return result;
}
```

Output:

```
Max value is: 200
```
[2]

2. C++ defines a function which also be referred as a subroutine, a procedure, a subprogram or even a method, carries out tasks defined by a sequence of instructions inside a statements block.A function must be declared before it can be used, with a name to identifying, type of return value, and type of any arguments passed to it. C++ calls this declaration "Function prototyping".

```
Example: intsubtraction(int, int);  //function prototyping
            . . .
      int subtraction(int n1, int n2){ //Function definition
            return (n1 - n2);
         }
```

In C++, if the body of a function contains a few lines of code, it can be coded "inline" and such function is called inline function. This type of inline function replaces function prototyping. In Object-Oriented Programming (OOP), functions defined in a class do not need prototyping.

```
Example: inline intsubtraction(int n1, int n2){return (n1 - n2)};
```
[3]

3. C#is the principal language in **.**NET (Dot Net) platform developed by Microsoft Corp. It was built on the lessons of other languages: C (high performance), C++ (object-oriented), Java (garbage collection, high security), and Visual Basic (rapid development). C# is regarded as a "100% Object-Oriented" language likes Java, even the most simple program which has nothing to do with object-oriented must be enclosed inside a class.

```
Example:   class SimpleProgram{
            static void Main(string[] args){
                  System.Console.WriteLine("Hello World!");
               }
           }
```
[4]

C# defines method is a basic part of a program. It can solve a certain problem, receiving parameters and return a result. We might say, methods are fundamental components of a program or a collection of methods makes up a program. In C#, the order of the methods is not important and method invocation is allowed before it is defined (no method prototyping) as in the example following.

Example:

```
static void Main(){       //Main method
      . . .
      PrintLogo();        //Call method PrintLogo
      . . .
}
static void PrintLogo(){  //Define method PrintLogo
Console.WriteLine("Microsoft Corp.");
}
```

[5]

III. 2. JAVA

Java is the current most popular programming language, we can see Java everywhere, from personal computers to other small application electronic devices such as cellphones, notepads, etc... In Java, when we write a stand-alone application, one of the classes in the file must has the same name with the name of the file, and also that class must contains the main method. [6]

Example:

```
public class TestJava{
      ...
public static void main(String[] args){
      ...
}
    ...
}
```

In the example above, the name of the file must be: TestJava.java

Most statements in Java are similar to the one in C# language. The ways they handle subprograms or methods should also be the same. The following are examples for illustration the use of methods in Java.

Example-1: Without method declaration (prototyping), and keyword "static"

```
public class TestMethods{
      public static void main(String[] args){ //main method
intfahrenheit = 107;
intcelsius = convertCelsius(fahrenheit);
display(fahrenheit, celsius);
      }
      publicintconvertCelsius(int f){       //convertCelsius method
      return (int)(5 * (f - 32) / 9);
      }
      public void display(int f, int c){      //display method
      System.out.printf("%5d%10d", f, c);
            System.out.println("\n");
      }
}
```

Compile the program above will get 2 errors because the two methods convertCelsius() and display() are not declared, be known to the compiler. Next, we declare the two methods to see what will happen?

Example-2: Declaring two subordinate methods by prototyping as in "C family"

```
public class TestMethods{
      publicintconvertCelsius(int f);  //method prototyping
      public void display(int f, int c); //method prototyping
      public static void main(String[] args){ //main method
intfahrenheit = 107;
intcelsius = convertCelsius(fahrenheit);
display(fahrenheit, celsius);
      }
      publicintconvertCelsius(int f){   //convertCelsius method
      return (int)(5 * (f - 32) / 9);
      }
      public void display(int f, int c){  //display method
      System.out.printf("%5d%10d", f, c);
      System.out.println("\n");
      }
}
```

Compile the program again, this time the errors because the methods were already defined above (prototyping). So we can say that Java does not allow method prototyping. Next, we place the subordinate methods before the main "up-side-down" as in the early Pascal.

Example-3: Place the subordinate methods before the main method

```
public class TestMethods{
      publicintconvertCelsius(int f){
      return (int)(5 * (f - 32) / 9);
      }
      public void display(int f, int c){
      System.out.printf("%5d%10d", f, c);
      System.out.println("\n");
      }
      public static void main(String[] args){
      intfahrenheit = 107;
      intcelsius = convertCelsius(fahrenheit);
      display(fahrenheit, celsius);
      }
}
```

The result is the same, the errors caused from non-static methods. So we need to put the keyword “static” back on the two subordinate methods, just likes the example in C# section.

Example-4: With keyword “static”

```
public class TestMethods{
      public static void main(String[] args){ //main method
intfahrenheit = 107;
intcelsius = convertCelsius(fahrenheit);
display(fahrenheit, celsius);
      }
      public static intconvertCelsius(int f){//convertCelsius method
      return (int)(5 * (f - 32) / 9);
      }
      public static void display(int f, int c){//display method
      System.out.printf("%5d%10d", f, c);
      System.out.println("\n");
      }
}
```

This time the program works with output:

```
  107        41

Press any key to continue . . .
```

In general, in object-oriented programming, methods declared inside a class do not need prototyping. We do not need to insert the keyword "static" in method header, these methods are classified as "instance methods" since they need an object to invoke them. Other methods are called "primitive" methods such as primitive datatypes.

IV. CONCLUSION

One of the problem solving technique that derived from the ancient Roman Empire era, "Divide and Conquer" which still holds today, and maybe true forever… When we have to deal with a complex problem, the best way to solve this such problem is to apply "Divide and Conquer" technique. We need to break the main (whole) problem into smaller sub problems or subprograms. Then, the problems become simpler, easy to find the solutions and to implement since each subprogram handles a single task. Finally, all solutions to subprograms are groupped together in a logical order to form the overall solution to the big and complex problem.

Depending on the programming language, subprograms are implemented slightly different form one to another, but the concept of using subprogram in programming or problem solving is still very much the same.

Reference:

[1] Sam A. Abolrous (2002), *Learn Pascal in Three Days (3e)*, Wordware Publishing, Inc., Plano, Texas, ISBN: 1-55622-805-8, pages:ix, 121, 122

[2] *C Programming Tutorial*. Tutorialspoint.com, pages: 60, 61

[3] Dirk Hunniger. *C++ Programming*. Wikibooks.org, pages: 229-231

[4] Jesse Liberty, Donald Xie (2008), *Programming C# 3.0, (5e)*. O'Reilly Media, Inc. Sebastopol, CA. 95472, ISBN: 10-696-52743-8, pages: x, 7

[5] Dilyan Dimitrov, et al, (2013). *Fundamentals of Computer Programming with C#*. Svetlin Nakov& Co., Sofia, Bulgary, ISBN: 978-954-400-773-7, pages: 293, 303

[6] Bruce Eckel (2006). *Thinking in Java (4e)*.Prentice Hall, Upper Saddle River, NJ. ISBN: 0-13-187248-6, pages: 53