DIbugger: User Manual

Benedikt Wagner Chiara Staudenmaier Etienne Brunner
Joana Plewnia Pascal Zwick Ulla Scheler

Betreuer: Mihai Herda, Michael Kirsten

January 31, 2018

Contents

Contents

Chapter 1

Quick Start

Chapter 2

The User Interface

Chapter 3

Configuring the DIbugger

3.1 Configuring the Language

You can change the language of the DIbugger with: Menu > Settings > Change language

3.2 Configuring the Maximum Number of Function Calls

The maximum number of function calls determines how many function calls are allowed within one program (including the obligatory main()-method). This number can be used to impose a limit on the depth of recursion within your program. You can change the maximum number of function calls with: Menu > Settings > Change maximum function calls

3.3 Configuring the Maximum Number of Iterations

The maximum number of iterations determines how many iterations of a while-loop are allowed within one program, thus guaranteeing that a loop cannot run forever. You can change the maximum number of iterations with: Menu > Settings > Change maximum iterations

Chapter 4

Loading and Saving

4.1 Adding a program

The file menu allows you to add a program to a new program window. You are asked to choose the file you want to open within the new program window. If you cancel the file choosing action, you are presented with an empty new program window. You can do this with:
Menu > File > Add a Program

4.2 Saving DIbugger-Configurations

Saving a DIbugger-Configuration means saving not only the code in your program windows but your Watch-Expressions, Breakpoints and Conditional Breakpoints, too. You can do this with:
Menu > File > Save config

4.3 Saving Code

There is no dedicated menu entry to save your program code as your code is automatically saved when you save your DIbugger-Configuration. You can do this with:
Menu > File > Save config

4.4 Loading DIbugger-Configurations

You can load a DIBugger-Configuration (that is all your program windows, Breakpoints, Conditional Breakpoints and Watch-Expressions) with: Menu > File > Load config

4.5 Removing a program

You can remove one of you programs as far as there are more than two programs. The button you need is the "X" Button in the right top of the program you want to remove.

Chapter 5

Editing Program Code

5.1 Resetting a Program Code Window

You can reset your programcode the same way as you remove a program described in **??**

5.2 Required Format of the Code

The format of the Wlang code is basically in C syntax, altough here are some differences explained above.

Writing functions and procedures You can write a routine in Wlang with the syntax `<type or void> <routinename>(<list of arguments>) <content of the routine>`. This is the same syntax as in the programming language C. For every program it is necessary, that there is a main-Routine with the name main. This is the entry point of the execution. All other routines have to be declared and implemented above.

Declaring arrays Array declaration has the syntax `<type> <dimensions> <arrayname>`.

Calling functions You can call a function in a assignment. Within this assignment the functioncall is the only thing on the right side e.g. `x = foo(y);`

5.3 Sample Program

As an easy example, see the implementation of the factorial of an integer in tow different ways.

```
//factorial programmed in an iterative manner
int main(int n){
    int i = 1;
```

```
    int sum = 1;
    while(i<=n) {
         sum = sum * i;
         i = i + 1;
    }
    return sum;
}


//other functions must be delcared before the main
int fac(int k) {
    //Calculate the factorial of k recursively.
    if (k <= 1)
         return 1;
    int res;
    res = fac(k-1); //this is the correct way to call functions.
    res = res * k;
    return res;
}

//every program needs a main method
int main(int k) {
    int res;
    res = fac(k);
    return res;
}
```

5.4 Common Mistakes

- You always need a main method. Without a correct main method a syntax error pop-up will occur.
- You cannot write a routine inside another routine. This leads to a syntax error, too.
- Make sure that all array declarations are in the form described in this paragraph. It is not allowed to write the dimensions after the name of the array.
- Make sure that all input values are seperated with a semikolon.
- It is not allowed to call a function and use its return value for calculations immediately, e.g. `x = bar(4)+3;`. The function call must stand alone on the right side of the assignment.

5.5 Auto-Generating Input

To get suggestions for input variables, you can use the menu as follows: Menu > Suggestions > Suggest Input Values

Chapter 6

Watch-Expressions and Conditional Breakpoints

Chapter 7

Debugging Program Code

After two or more programs are added you can start the Debug-Mode with the green play button in the top right corner of the DIbugger. More general: to control the debugging process, use the buttons on the right. It is always possible to stop the Debug-Mode and return to Edit-Mode by clicking the red square stop button beside the play button.

7.1 Making Steps

There are different buttons for different kinds of steps.

Step The button labeled with "step" leads to an execution of all programs according to the number of commands given in the specified stepsize. As an example, assume the stepsize of program A is 3 and the stepsize of program B is 2. Then there are 3 commands executed in program A and 2 commands in program B. Encountering a function call, the execution is always steping into the function.

StepBack The button labeled with "stepBack" causes the DIbugger to rewind one command in each program.

StepOut The button labeled with "stepOut" causes the DIbugger to jump out of the current function in every program.

StepOver The button labeled with "stepOver" causes the DIbugger to jump over the next command, if it is a function call.

Continue The button labeled with "continue" causes the DIbugger to run each program until either a conditional breakpoint evaluates to true or a normal breakpoint occurs.

Singlestep There is one button up above each program with the label "singlestep". This button will run exactly one command only in the program it belongs to.

7.2 Understanding the Output

You can inspect the variables occuring in a program in the variable inspector, which is settled under the programs code. With your right mouse button you can hide a variable.
A program, whose main-routine has a return type i.e. is a function, will produce an output. This return value will be printed right under the variable inspector.