

Dlbugger: User Manual

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

Betreuer: Mihai Herda, Michael Kirsten

January 31, 2018

Contents

Contents

1	Quick Start	1
2	The User Interface	3
3	Configuring the DDebugger	5
3.1	Configuring the Language	5
3.2	Configuring the Maximum Number of Function Calls	5
3.3	Configuring the Maximum Number of Iterations	5
4	Loading and Saving	7
4.1	Adding a program	7
4.2	Saving DDebugger-Configurations	7
4.3	Saving Code	7
4.4	Loading DDebugger-Configurations	7
4.5	Removing a program	8
5	Editing Program Code	9
5.1	Resetting a Program Code Window	9
5.2	Required Format of the Code	9
5.3	Sample Program	9
5.4	Common Mistakes	10
5.5	Auto-Generating Input	10
6	Watch-Expressions and Conditional Breakpoints	11
6.1	Adding and Changing Watch-Expressions	11
6.2	Adding and Changing Conditional Breakpoints	11
6.3	Auto-Generating Watch-Expressions and Conditional Breakpoints	11
7	Debugging Program Code	13
7.1	Making Steps	13
7.2	Understanding the Output	13

Chapter 1

Quick Start

Chapter 2

The User Interface

Chapter 3

Configuring the DDebugger

3.1 Configuring the Language

You can change the language of the DDebugger 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, although 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 declared 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

5.5 Auto-Generating Input

Chapter 6

Watch-Expressions and Conditional Breakpoints

- 6.1 Adding and Changing Watch-Expressions
- 6.2 Adding and Changing Conditional Breakpoints
- 6.3 Auto-Generating Watch-Expressions and
Conditional Breakpoints

Chapter 7

Debugging Program Code

7.1 Making Steps

7.2 Understanding the Output