Efficient Debugging

CPRE 416-Software Evolution and Maintenance-Lecture 11

Contents

- Debugging challenge
- Use of program slicing
- Static vs. dynamic slicing
- Reaching definitions
- Program slice browser

Debugging Challenge

- Debugging is a complex and difficult activity.
- The failure may be manifested far away from the fault (bug) itself.
- Analysis is needed determine the cause and the location of a program failure.
- Analysis is often tedious and time consuming because of many dependencies and many execution paths

Sample Program

- The program computes the sum of the areas of N triangles.
- It reads the value of N, followed by the lengths of the three sides of each of these N triangles.
- It classifies each triangle as an equilateral, isosceles, right, or a scalene triangle. Then it computes the area of the triangle using an appropriate formula.
- Finally, the program prints the sum of the areas.

```c
/* Find the sum of areas of given triangles. */
#define MAX 100
typedef enum {equilateral, isosceles, right, scalene} class_type;
typedef struct {int a, b, c;} triangle_type;

main()
{
    triangle_type sides[MAX];
    class_type class;
    int a_sqr, b_sqr, c_sqr, N, i;
    double area, sum, s, sqrt();

    printf("Enter number of triangles:\n");
    scanf("%d", &N);
    for (i = 0; i < N; i++) {
        printf("Enter three sides of triangle %d in ascending order:\n", i+1);
        scanf("%d %d %d", &sides[i].a, &sides[i].b, &sides[i].c);
    }

    sum = 0;
    i = 0;
    while (i < N) {
        a_sqr = sides[i].a * sides[i].a;
        b_sqr = sides[i].b * sides[i].b;
        c_sqr = sides[i].c * sides[i].c;
        if ((sides[i].a == sides[i].b) && (sides[i].b = sides[i].c))
            class = equilateral;
        else if ((sides[i].a == sides[i].b) || (sides[i].b == sides[i].c))
            class = isosceles;
        else if (a_sqr == b_sqr + c_sqr)
            class = right;
        else class = scalene;

        if (class == right)
            area = sides[i].b * sides[i].c / 2.0;
        else if (class == equilateral)
            area = sides[i].a * sides[i].a * sqrt(3.0) / 4.0;
        else {
            s = (sides[i].a + sides[i].b + sides[i].c) / 2.0;
            area = sqrt(s * (s - sides[i].a) * (s - sides[i].b) *
                   (s - sides[i].c));
        }
        sum += area;
        i += 1;
    }
    printf("Sum of areas of the %d triangles is %.2f.\n", N, sum);
}
```


Start Debugging

- Suppose this program is executed for the test case #1 when N = 2 and sides of the two triangles are (5, 4, 3) and (4, 4, 2) respectively.
- If the final sum of areas printed is incorrect, how should we go about locating the bug in the program?

Many Possibilities for Bugs

- Looking backwards from the **printf** statement on line 46, there are several possibilities:
  - sum is not being updated properly
  - one or more of the formulas for computing the area of a triangle are incorrect
  - the triangle is being classified incorrectly
  - the values for the three sides of the triangle are not being read correctly.

Debugging Steps

1. Determine which statements in the code have an influence on the value of sum at line 46.
2. Select one (or more) of these statements at which to examine the program state.
3. Recreate the program state at those statements to examine specific variables.

Program Slicing

- It is used to find all those statements in a program that directly or indirectly affect the value of a given variable occurrence (variable name and statement location).
- The statements that affect the value constitute the *slice* of the program with respect to the given variable occurrence

Two Types of Program Slices

- Two types of slices: *static* and *dynamic*.
- A static slice includes all statements that *could* influence the value of a variable occurrence for *all* possible inputs to the program.
- A dynamic slice includes only those statements that influence the value of a variable occurrence for a given test case.

Data vs. Control Slices

- A data slice is one that is defined with respect to a data value—a variable occurrence.
  - Useful when we are trying to determine the source of a wrong value for a variable occurrence.
- A control slice is one that is defined with respect to control reaching a certain program location.
  - Useful when we are trying to determine why the control has reached a wrong location.

Reaching Definitions

- Looking at the whole slice at once may be overwhelming.
- It may be useful to analyze the inter-statement program dependencies one at a time – incremental slicing.
- Reaching definitions are the assignments (or modifications) of variables that reach and affect the given variable occurrence.

Terminology

- Def: a *definition* of a variable x is a statement that assigns a value to x
- Use: a *use* of a variable x is a statement that reads the value of x
- Kill: a definition of a variable x on a path is said to be *killed* on a control flow path if we encounter another definition of variable.
- Reaches: a definition *d reaches* a point *p* if there is a path from the point immediately following *d* to *p* - such that *d* is not killed along *p*.

Example

```
1:   I :=  M - 2;
2:   J :=  N;
3:   A:= U1;
4:
5:   DO
6:      I  :=   I +1;
7:      J :=   J-1;
8:       if (E1)   then
9:                A:=   A + 2;
10:     else
11:               I :=  u3;
12:     endif
13:  WHILE  e2
```

- def of A at 3 and 9
- The definition at line 9 kills the definition of A from line 3
- The definition of A from line 3 reaches line 9

Program Slice Browser

- We developed a program slice browser (PSB) to provide the user a visual representation of the slice.
- Using the PSB:
  - A slice can be viewed at different levels of granularity: statements, control blocks, and procedures.
  - By coupling the visual representation and the corresponding source one can conveniently navigate through the code.

References

- Efficient debugging with slicing and backtracking: http://www2.umassd.edu/SWPI/slicing/purdue/TR80P.pdf
- Program slice browser: http://doi.ieeecomputersociety.org/10.1109/WPC.2001.921713
-

