DEBUG

Make Life Beautiful Again

@zccz14

References

- 《调试九法：软硬件错误的排查之道》
  Debugging: The 9 Indispensable Rules for Finding Even the Most Elusive Software and Hardware Problems
- 《提问的智慧》
  How To Ask Questions The Smart Way

BUG

- A software bug is an error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

  — Wikipedia

- Keywords: incorrect, unexpected, unintended

Compiler vs IDE

Compiler

- Compiler is a program which translates high-level source code to low-level target code.
- e.g. gcc, msvc, clang, javac…

IDE

- IDE(Integrated Dev Env)
- IDE = Editor + Compiler + Runtime Env + Debugger + …
- e.g. Visual Studio, Dev-C++, Intellij IDEA, Eclipse…

Recognition

What happened? Recognize which errors occurred first.

Type::Error

- Compilation Error (CE)
- Link Error (LE, LNK)
- Runtime Error (RE)

- *Logic Error

Type::Error::CE

- Lexical Error
  e.g. error: invalid suffix "a" on integer constant.
- Grammar Error
  e.g. error: expected ',' or ';' before '}' token.
- Semantic Error
  e.g. error: 'a' was not declared in this scope.
- Internal Error
  Compiler bug, not your fault.

Avalanche Errors

Errors may cause secondary errors.

```
#include <iostream>
int main(){
  int a = 10
  for (int i = 0; i < a; i++) {
    std::cout << i << std::endl;
  }
  return 0;
}
```

Type::Error::LE

- Write Rejection

  e.g. ld.exe: cannot open output file a.exe: Permission denied

  collect2.exe: error: ld returned 1 exit status

- Undefined Reference

  e.g. undefined reference to 'a'.

- Multiple Definition

  e.g. multiple definition of 'main'.

Type::Error::RE

"The program has stopped working."

The return value of process indicates the type of runtime error.

- 0x00000000: OK
- 0xC0000005: ACCESS_VIOLATION (-1073741819)
- 0xC0000094: INT_DIVIDED_BY_ZERO (-1073741676)

Gathering

Where are the bugs? Collect enough information first.

Info Gathering Techniques

- Static analysis (hard)
- Printing intermediate values (median)
- Runtime tracking (easy)

Static Analysis

Analyze the source code without running.

- Hard but basic.
- Understand the whole system.
- Time cost highly depends on experience.

Print Intermediate Values

Based on static analysis, print the doubtful intermediate values. In order to validate your worrying.

- Call output functions inline. e.g.
  - printf in C
  - std::cout in C++
  - System.out.println in Java
  - Console.WriteLine in C#
  - console.log in JavaScript
  - …

Debug Macro in C (stdio)

- Preparation:

```
#define Debug(x, f) printf(#x" = "#f"\n", x)
```

- Usage:

```
int cube(int x) {
  // printf("x * x = %d\n", x * x);
  Debug(x * x, %d);
  return x * x * x;
}
```

- Output: (for x = 9)

```
x * x = 81
```

Debug Macro in C++ (stream)

- Preparation:

```
#define Debug(x) #x << " = " << x
```

- Usage:

```
int cube(int x) {
  // cout << "x * x" << " = " << x * x << endl;
  std::cout << Debug(x * x) << std::endl;
  return x * x * x;
}
```

- Output: (for x = 9)

```
x * x = 81
```

And then…

```
int f(int n){
  Print(n);
  int sum = 0;
  Print(sum);
  while (n--) {
    sum += n;
    Print(n);
    Print(sum);
  }
  Print(sum);
  return sum;
}
// overdo!
```

Runtime Tracking

Run the program and track it.
Setup break-point to make the program stop.

- Easy to find bug.
- Too much redundant information.

Code Format

Not only pretty style, but also design intention.

Avoid Logic Error 1

Unformatted

```
int f(int x){
  int sum = 0;
  while (x)
    sum += x; x /= 2;
  return sum;
}
```

formatted

```
int f(int x) {
  int sum = 0;
  while (x)
    sum += x;
  x /= 2;
  return sum;
}
```

Avoid Logic Error 2

Unformatted

```
int f(int n){
  int sum = 0, i;
  for (i = 0; i < n; i++);
    sum += i;
  return sum;
}
```

formatted

```
int f(int n) {
  int sum = 0, i;
  for (i = 0; i < n; i++)
    ;
  sum += i;
  return sum;
}
```

Avoid Logic Error 3

Unformatted

```
int f(int n){
  int res = n;
  if (n > 0)
    if (n > 100)
      res = 100;
  else
    res = 0;
  return res;
}
```

Formatted

```
int f(int n){
  int res = n;
  if (n > 0)
    if (n > 100)
      res = 100;
    else
      res = 0;
  return res;
}
```

Code Format Tools

- clang-format

  clang-format is a format tool based on LLVM.

- Visual Studio

  There is an embedded reformat tool (code style inspector).

- Astyle

  Old code formatter, embedded in Dev-C++.

After locked on bugs…

- If you know how to fix it, just do it.
  - Don’t forget the regression testing!
- Else…
  - Google is your friend
  - RTFM & STFW
  - Ask someone for help

Samples

Real stories about debug.

#1 String Intercross

Write a program to intercross 2 strings.
`void intercross(char s1[], char s2[], char s3[]);`

For Examples:
`char s1[] = "abc", s2[] = "fgh";`

You are ought to let s3 equals to "afbgch".
Ensure that s1 and s2 are not modified.

#2 Series Calculus

Write a program to figure out arctan(x) with following equation:

$$\tan^{-1} x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}$$

The absolute error should less than 10^{-8}.

#3 Black Hole Number

Define an operation on k-digit integer:

1. Split each digit into an integer array `'a'`;
2. Descending sort `'a'` and join each digit into an integer `'n1'`;
3. Reverse sorted array `'a'` and join each digit into an integer `'n2'`;
4. Calculate the result = `'n1' - 'n2'`;

Repeat the operation until arrive fixed point.
For the given integer, figure out the corresponding fixed point.

#1 String Intercross

Write a program to intercross 2 strings.

```
void intercross(char s1[], char s2[], char s3[]);
```

For Examples:

```
char s1[] = “abc”, s2[] = “fgh”;
```

You are ought to let s3 equals to “afbgch”.

Ensure that s1 and s2 are not modified.

Someone's Implementation

```
void intercross(char s1[],char s2[],char s3[]){
  int i, j;
  for(i = 0; s1[i] != '\0'; i++)
    s3[i * 2] = s1[i];
  for(j = 0; s2[j] != '\0'; j++)
    s3[j * 2 + 1] = s2[j];
}
// What's the matter?
```

Someone's Implementation

```
#include <stdio.h>
void intercross(char [], char [], char []);
int main(){
  char s1[100], s2[100], s3[100];
  gets(s1);
  gets(s2);
  intercross(s1, s2, s3);
  puts(s3);
}
```

Simple Patch

```
#include <stdio.h>
void intercross(char[], char[], char[]);
char s1[100], s2[100], s3[100];
int main() {
  gets(s1);
  gets(s2);
  intercross(s1, s2, s3);
  puts(s3);
}
// Just move out the string declaration!
// You can pass sample input & output. But why?
```

What if …?

- What if strlen(s1) != strlen(s2)?
- What if s3 is not zero-initialized?
- What if access index is out of range?
- What if s1 and s2 are not zero-terminated?

Additional Specification

For the function ‘intercross’:

- When strlen(s1) != strlen(s2), flush the remaining string to s3.
  - s1 = “abcd”, s2 = “ef” => s3 = “aebfcd”.

Invoking Rules

- Caller should allocate enough memory for s1, s2 and s3.
  - s3 need *strlen(s1) + strlen(s2) - 1* bytes at least.
- Caller should ensure that s1 and s2 are zero-terminated.
- Callee should ensure that s3 is zero-terminated.

Final Callee

```
void intercross(char s1[], char s2[], char s3[])
{
  while (*s1 || *s2) {
    if (*s1) *s3++ = *s1++;
    if (*s2) *s3++ = *s2++;
  }
  *s3 = 0;
}
// WTF???
```

Friendly Callee

```
void intercross(char s1[], char s2[], char s3[]) {
  int i = 0, j = 0, k = 0;
  while (s1[i] != '\0' || s2[j] != '\0') {
    if (s1[i] != '\0') {
      s3[k] = s1[i];
      k++; i++;
    }
    if (s2[j] != '\0') {
      s3[k] = s2[j];
      k++; j++;
    }
  }
  s3[k] = '\0';
}
```

General Procedure

Take a break to think about the general debug procedure.

General Procedure

1. Confirm that there are bugs.
2. Take it easy and classify the symptoms.
3. Collect enough information to find out where the bugs are.
4. Think about how to defeat them.
5. Log your story & enhance your debug skill.

#2 Series Calculus

Write a program to figure out arctan(x) with following equation:

$$\tan^{-1} x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}$$

The absolute error should less than 10^{-8}.

Someone's implementation

```
double arctan(double x) {
  double res = x, xx = x;
  int n = 0;
  while (fabs(res - atan(x)) > 1e-8) {
    n++;
    xx *= -(x * x * (2 * n - 1) / (2 * n + 1));
    res += xx;
  }
  return res;
}
// What's the matter?
```

Print Intermediate Values

```
double arctan(double x) {
  double res = x, xx = x;
  int n = 0;
  while (fabs(res - atan(x)) > 1e-8) {
    printf("%d %e %e\n", n, res, xx);
    n++;
    xx *= -(x * x * (2 * n - 1) / (2 * n + 1));
    res += xx;
  }
  return res;
}
```

arctan(1.2) = 0.876058

n	res	xx
0	1.2	1.2
1	0.624	-0.576
10	2.117829	2.190720
100	2.4e+13	4.1e+13
1000	8.151189e+154	1.381740e+155
1970	INF	INF

Result: The algorithm is incorrect for x > 1!

#2 Series Calculus*

Write a program to figure out arctan(x) with following equation:

$$\tan^{-1} x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}, (|x| < 1)$$

The absolute error should less than 10^{-8}.

Someone's implementation

```
double arctan(double x) {
  double res = x, xx = x;
  int n = 0;
  while (fabs(res - atan(x)) > 1e-8) {
    n++;
    xx *= -(x * x * (2 * n - 1) / (2 * n + 1));
    res += xx;
  }
  return res;
}
// What's the matter?
```

Final Implementation

```
double arctan(double x) {
  double res = x, xx = x;
  int n = 0;
  while (fabs(xx) > 1e-8) {
    n++;
    xx *= -(x * x * (2 * n - 1) / (2 * n + 1));
    res += xx;
  }
  return res;
}
```

#3 Black Hole Number

Define an operation on *k*-digit integer:

1. Split each digit into an integer array *'a'*;
2. Descending sort 'a' and join each digit into an integer 'n1';
3. Reverse sorted array 'a' and join each digit into an integer 'n2';
4. Calculate the result = 'n1' - 'n2';

Repeat the operation until arrive fixed point.

For the given integer, figure out the corresponding fixed point.

Given function prototype

- `int digits(int n);`
- `void split(int a[], int n, int k);`
- `void sortd(int a[], int k);`
- `void reverse(int a[], int k);`
- `int combine(int a[], int k);`

Given main function

```
int main() {
  int oldn = -1, n, a[10], n1, n2;
  scanf("%d", &n);
  int k = digits(n);
  while (n != oldn) {
    oldn = n;
    split(a, n, k);
    sortd(a, k);
    n1 = combine(a, k);
    reverse(a, k);
    n2 = combine(a, k);
    n = n1 - n2;
    printf("%d-%d=%d\n", n1, n2, n);
  }
}
```

Someone's Implementation

```
int digits(int n) {
  int i = 0;
  if (n != 0)
    n = n / 10;
      i++;
  return i;
}
// What's the matter?
```

Simple Patch

```
int digits(int n) {
  int i = 0;
  while (n != 0) {
    n = n / 10;
    i++;
  }
  return i;
}
```

Someone's Implementation

```
void split(int a[], int n, int k) {
  int i;
  for (i = k - 1; i >= 0; i--)
    while(n != 0) {
      a[i] = n % 10;
      n /= 10;
    }
}
// What's the matter?
```

Simple Patch

```
void split(int a[], int n, int k) {
  int i;
  for (i = k - 1; i >= 0; i--) {
    // while(n != 0) {
    a[i] = n % 10;
    n /= 10;
    // }
  }
}
```

Someone's Implementation

```
void sortd(int a[], int k) {
  int i, j, t;
  for (i = 0; i < k - 1; i++)
    for (j = 0; j < k - 1 - i; j++)
      if (a[j] < a[j + 1]) {
        t = a[j];
        a[j] = a[j + 1];
        a[j + 1] = t;
      }
}
// What's the matter?
```

Someone's Implementation

```
void reverse(int a[], int k) {
  int i, j, t;
  for (i = 0, j = k - 1; i < j; i++, j--) {
    t = a[i];
    a[i] = a[j];
    a[j] = t;
  }
}
// What's the matter?
```

Someone's Implementation

```
int combine(int a[], int k) {
  int i, sum = 0;
  for(i = k - 1; I >= 0; i--)
    sum *= 10;
    sum += a[i];
  return sum;
}
// What's the matter?
```

Simple Patch

```
int combine(int a[], int k) {
  int i, sum = 0;
  for (i = 0; i < k; i++)
    sum = 10 * sum + a[i];
  return sum;
}
```

Get debugging advice

- 1. paste your code with your design intention top commented to https://paste.ubuntu.com/. And you will get an URL like https://paste.ubuntu.com/24286112/
- 2. leave the URL to
  - GitHub Issue: https://github.com/zccz14/QA/
  - Bilibili Live ID: 1701830 (Danmaku)
  - Send Email: zccz14@outlook.com
- 3. wait for response.

  Min patch and probably writing suggestions.