Отладка и тестирование

Ошибки в программе есть всегда

Тестирование — выполнение программы с целью обнаружения ошибок

Отладка — определение места ошибки и внесение исправлений в программу

Сбой и ошибка

Сбой — неверная работа программы

Ошибка — место в программе, где написан некорректный код

```
1  a = 0;
2  if (x > 3)
3     a = 10;
4  b = 100 / a;
```

Тестирование

Тестирование — выполнение программы на некотором наборе данных с целью поиска ошибок

Тест

- Входные данные
- Ожидаемое поведение программы (выходные данные)

С разными входными данными работают разные куски программы.

Критерии тестирования

Критерий покрытия путей

Нужно покрыть тестами как можно больше путей в программе.

```
1  if (a == 0)
2    std::cout << "1";
3  if (b == 0)
4    std::cout << "2";
5  if (c == 0)
6    std::cout << "3";
```

Критерии тестирования

Критерий покрытия путей

Тесты

- вход: $a = 0, b = 0, c = 0,$ выход: 123

Критерии тестирования

Критерий покрытия путей

Тесты

- вход: $a = 0, b = 0, c = 0,$ выход: 123
- вход: $a = 0, b = 0, c = 1,$ выход: 12

Критерии тестирования

Критерий покрытия путей

Тесты

- вход: $a = 0, b = 0, c = 0,$ выход: 123
- вход: $a = 0, b = 0, c = 1,$ выход: 12
- вход: $a = 0, b = 1, c = 0,$ выход: 13

Критерии тестирования

Критерий покрытия путей

Тесты

- вход: $a = 0, b = 0, c = 0$, выход: 123
- вход: $a = 0, b = 0, c = 1$, выход: 12
- вход: $a = 0, b = 1, c = 0$, выход: 13
- вход: $a = 0, b = 1, c = 1$, выход: 1

Критерии тестирования

Критерий покрытия путей

Тесты

- вход: $a = 0, b = 0, c = 0$, выход: 123
- вход: $a = 0, b = 0, c = 1$, выход: 12
- вход: $a = 0, b = 1, c = 0$, выход: 13
- вход: $a = 0, b = 1, c = 1$, выход: 1
- вход: $a = 1, b = 0, c = 0$, выход: 23

Критерии тестирования

Критерий покрытия путей

Тесты

- вход: $a = 0, b = 0, c = 0$, выход: 123
- вход: $a = 0, b = 0, c = 1$, выход: 12
- вход: $a = 0, b = 1, c = 0$, выход: 13
- вход: $a = 0, b = 1, c = 1$, выход: 1
- вход: $a = 1, b = 0, c = 0$, выход: 23
- вход: $a = 1, b = 0, c = 1$, выход: 2

Критерии тестирования

Критерий покрытия путей

Тесты

- вход: $a = 0, b = 0, c = 0$, выход: 123
- вход: $a = 0, b = 0, c = 1$, выход: 12
- вход: $a = 0, b = 1, c = 0$, выход: 13
- вход: $a = 0, b = 1, c = 1$, выход: 1
- вход: $a = 1, b = 0, c = 0$, выход: 23
- вход: $a = 1, b = 0, c = 1$, выход: 2
- вход: $a = 1, b = 1, c = 0$, выход: 3

Критерии тестирования

Критерий покрытия путей

Тесты

- вход: $a = 0, b = 0, c = 0$, выход: 123
- вход: $a = 0, b = 0, c = 1$, выход: 12
- вход: $a = 0, b = 1, c = 0$, выход: 13
- вход: $a = 0, b = 1, c = 1$, выход: 1
- вход: $a = 1, b = 0, c = 0$, выход: 23
- вход: $a = 1, b = 0, c = 1$, выход: 2
- вход: $a = 1, b = 1, c = 0$, выход: 3
- вход: $a = 1, b = 1, c = 1$, выход:

Критерии тестирования

Критерий покрытия путей

Нужно покрыть тестами как можно больше путей в программе.

```
1  while (a != 0)
2  {
3      int b;
4      std::cin >> b;
5      a += b;
6  }
```

Критерии тестирования

Критерий покрытия ветвей

Нужно покрыть тестами как можно больше ветвей программы.

```
1  if (a == 0)
2    std::cout << "1";
3  if (b == 0)
4    std::cout << "2";
5  if (c == 0)
6    std::cout << "3";
```

Критерии тестирования

Критерий покрытия ветвей

```
1  if (a == 0)
2    std::cout << "1";
3  else
4    std::cout << "-1";
5  if (b == 0)
6    std::cout << "2";
7  else
8    std::cout << "-2";
9  if (c == 0)
10   std::cout << "3";
11 else
12   std::cout << "-3";
```

Критерии тестирования

Критерий покрытия условий

Чтобы каждая часть условия была покрыта хотя бы одним тестом.

```
1  if (a != 0 || c != 0)
2    std::cout << c / a;
3  else
4    std::cout << "0";
```

Как писать тесты

- Граничные значения входных данных

Как писать тесты

- Граничные значения входных данных
- Особенные значения для алгоритма (например, деление на 0)

Как писать тесты

- Граничные значения входных данных
- Особенные значения для алгоритма (например, деление на 0)
- Разные варианты вывода (например, «YES» и «NO»)

Как писать тесты

- Граничные значения входных данных
- Особенные значения для алгоритма (например, деление на 0)
- Разные варианты вывода (например, «YES» и «NO»)
- Максимальное время выполнения программы

Как писать тесты

- Граничные значения входных данных
- Особенные значения для алгоритма (например, деление на 0)
- Разные варианты вывода (например, «YES» и «NO»)
- Максимальное время выполнения программы
- Максимальное потребление памяти

Как писать тесты

- Граничные значения входных данных
- Особенные значения для алгоритма (например, деление на 0)
- Разные варианты вывода (например, «YES» и «NO»)
- Максимальное время выполнения программы
- Максимальное потребление памяти
- Несколько типичных тестов

Как писать тесты

- Граничные значения входных данных
- Особенные значения для алгоритма (например, деление на 0)
- Разные варианты вывода (например, «YES» и «NO»)
- Максимальное время выполнения программы
- Максимальное потребление памяти
- Несколько типичных тестов
- Несколько случайных тестов

Как писать тесты

- Граничные значения входных данных
- Особенные значения для алгоритма (например, деление на 0)
- Разные варианты вывода (например, «YES» и «NO»)
- Максимальное время выполнения программы
- Максимальное потребление памяти
- Несколько типичных тестов
- Несколько случайных тестов
- Желательно выполнить один из критериев покрытия тестами

Журнал тестировщика

N	Входные данные	Ожидаемые выходные данные	Наблюдаемые выходные данные
1	1 2 3 4	YES	NO
2	1 2 3 5	YES	
3			
4			
5			
6			

Принципы отладки

- Обдумывать результаты каждого теста

Принципы отладки

- Обдумывать результаты каждого теста
- Не вносить случайные исправления

Принципы отладки

- Обдумывать результаты каждого теста
- Не вносить случайные исправления
- Проверять тестами конкретные гипотезы

Принципы отладки

- Обдумывать результаты каждого теста
- Не вносить случайные исправления
- Проверять тестами конкретные гипотезы
- Исправлять ошибки по очереди

Принципы отладки

- Обдумывать результаты каждого теста
- Не вносить случайные исправления
- Проверять тестами конкретные гипотезы
- Исправлять ошибки по очереди
- Постараться объяснить все видимые симптомы сбоя

Принципы отладки

- Обдумывать результаты каждого теста
- Не вносить случайные исправления
- Проверять тестами конкретные гипотезы
- Исправлять ошибки по очереди
- Постараться объяснить все видимые симптомы сбоя
- Исправление может внести новые ошибки

Отладочные средства

- Отладочная печать

Отладочные средства

- Отладочная печать
- Интерактивные отладчики

Отладочные средства

- Отладочная печать
- Интерактивные отладчики
- Внешние трассировщики и профилировщики

Трассировка

Трассировка — это получение трассы работы программы.
Трасса может включать значения переменных.

Отладочный вывод

__LINE__ — номер строки
__FILE__ — номер файла

```
1  std::cout << "Line: " << __LINE__
2       << " in file " << __FILE__ << "\n";
```

Отладочный вывод

Для быстрой сортировки

```
1  void print(int line, int *a, int len)
2  {
3      std::cout << "L" << line << " ";
4      for (int i = 0 ; i < len ; ++i)
5          std::cout << a[i] << " ";
6      std::cout << "\n";
7  }
```

Источник значения

```
1  a = 1;
2  b = 2;
3  c = 3;
4  d = a + b;
5  e = c + b;
6  f = a + d;
```

```
1  a = 1;
2  b = 2;
3
4  d = a + b;
5
6  f = a + d;
```

Интерактивные отладчики

- Пошаговое исполнение
- Исполнение «до курсора»
- Просмотр значений переменных
- Точки останова (breakpoints)
- Точки трассировки (tracepoints)
- Точки наблюдения (watchpoints)
- Наблюдение за стеком