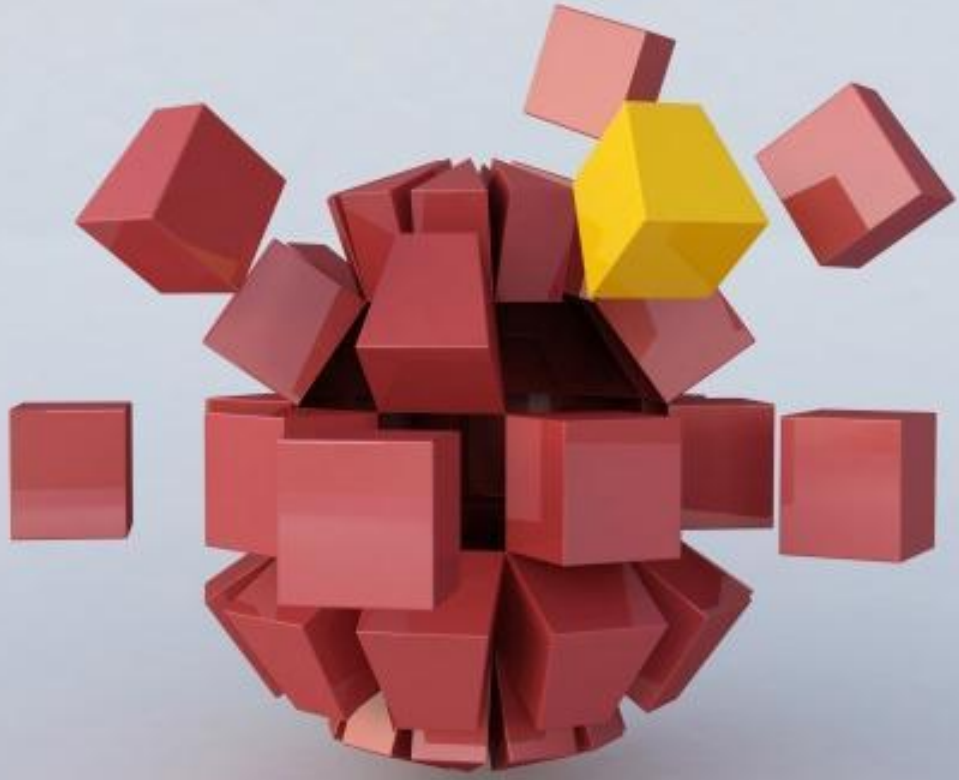
Понятие об ошибке программного обеспечения. Источники ошибок



Вопросы для повторения:

- Назовите некоторые советы по отладке программы
- Для чего необходимы комментарии в программе? Что необходимо комментировать в коде программы?

- Как определить, когда следует остановить отладку?
- Ясно, что отладка должна идти до тех пор, пока не будут выявлены все ошибки, — или нам так покажется.
- А как узнать, что мы нашли последнюю ошибку? Мы не знаем.
- Последняя ошибка — это шутка программистов. Такой ошибки не существует. В большой программе никогда невозможно найти последнюю ошибку.
- Кроме отладки, нам необходим систематический подход к поиску ош

Классификация ошибок:

1) Ошибки во время компиляции. Это ошибки, обнаруженные компилятором. Их можно подразделить на категории в зависимости от того, какие правила языка он нарушают:

- синтаксические ошибки;
- ошибки, связанные с типами.

Классификация ошибок:

2) Ошибки во время редактирования связей. Это ошибки, обнаруженные редактором связей при попытке объединить объектные файлы в выполняемый модуль.

Классификация ошибок:

3) Ошибки во время выполнения. Это ошибки, обнаруженные в ходе контрольных проверок выполняемого модуля. Эти ошибки подразделяются на следующие категории:

- ошибки, обнаруженные компьютером (аппаратным обеспечением и/или операционной системой);
- ошибки, обнаруженные с помощью библиотеки (например, стандартной);

Классификация ошибок:

4) Логические ошибки. Это ошибки, найденные программистом в поисках причины неправильных результатов.

Программа должна стремиться удовлетворять следующим условиям:

Часть основных проф. требований

1. Должна вычислять желаемые результаты при всех допустимых входных данных.
2. Должна выдавать осмысленные сообщения обо всех неправильных входных данных.
3. Не обязана обрабатывать ошибки аппаратного обеспечения.
4. Не обязана обрабатывать ошибки программного обеспечения.
5. Должна завершать работу

Источники ошибок:

- *Плохая спецификация.* (Плохо представили назначение программы → невозможно предусмотреть обработку всех ошибок)
- *Неполные программы.* В ходе разработки неизбежно возникают варианты, которые мы не предусмотрели.
- *Непредусмотренные аргументы.* Если функция принимает аргумент, который не был предусмотрен, то возникнет проблема (sqrt(-1.2)).
- *Непредусмотренные входные данные.*
- *Неожиданное состояние.* Большинство программ хранит большое количество данных ("состояний"): списки адресов, каталоги телефонов и данные о температуре, записанные в объекты типа vector. Что произойдет, если эти данные окажутся неполными или неправильными?
- *Логические ошибки.* Эти ошибки приводят к тому, что программа просто делает не то, что от нее ожидается.
- И др.

Ошибки во время компиляции

- Многие ошибки, которые обнаруживает компилятор, относятся к категории "грубых ошибок", представляющих собой ошибки, связанные с типами, или результат неполного редактирования кода. Другие ошибки являются результатом плохого понимания взаимодействия частей нашей программы.

Синтаксические ошибки

- //функция
- int area(int length, int width); // $S_{прямоуг.}$

- int si = area(7,2;
- int si = area(7)
- Int s3 = area(7);
- int s4 >> area(‘7,2’);

Синтаксические ошибки

- Итак, если вы не видите ничего неправильного в строке, на которую ссылается компилятор, проверьте **предшествующие** строки программы.

Ошибки, связанные с типами

- После того как вы устраните синтаксические ошибки, компилятор начнет выдавать сообщения об ошибках, связанных с типами переменных, функций и др.

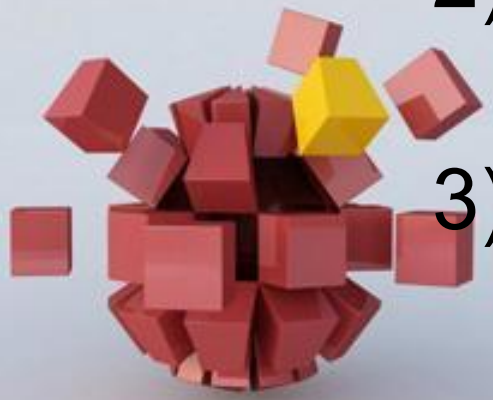
```
//функция
int area(int length, int width);
// S прямоуг.

int x0 = arena(7,2);
int x1 = area(7);
int x2 = area("seven",2) ;
```

Не ошибки
(логические ошибки)

- int area(int length, int width);
- // $S_{прямоуг.}$

1) int x4 = area(10,-7); // ОК: но …
2) int x5 = area(10.7,9.3); // ОК: но…
3) char x6 = area (100, 9999); // ОК: но …



Ошибки во время редактирования связей

- Любая программа состоит из нескольких отдельно компилируемых частей, которые называют единицами трансляции (translation units). Каждая функция в программе должна быть объявлена с теми же самыми типами, которые указаны во всех единицах трансляции, откуда она вызывается. Для этого используются заголовочные файлы. Кроме того, каждая функция должна быть объявлена в программе только один раз. Если хотя бы одно из этих правил нарушено, то редактор

Ошибки во время выполнения программы

- Если программа не содержит ошибок, которые можно обнаружить на этапах компиляции и редактирования связей, то она выполняется. Здесь-то и могут проявиться настоящие ошибки.
- Например, файл не найден.

Сообщения об ошибках

- Иногда можно вернуть сообщение «Неправильное значение».

Предотвратить ошибку при вызове функции area (x, y) в модуле main() относительно просто:

- `if (x<=0) error("неположительное x");`
- `if (y<=0) error("неположительное y");`
- `int area1 = area(x,y);`

Сообщения об ошибках

- Существует другой способ решить описанную проблему: использовать исключения (exceptions)

```
if(x==1)
// правильно!
{
   y=x+3;
   z=y*5;
}
```

Найдите ошибки:

```
if (x=1)
// неправильно!
// выполняется всегда!
{
    y=x+3;
    z=y*5;
}
```

Найдите ошибки:

```
if (x==1);
{
    y=x+3;
    z=y*5;
}
```

```
// неправильно!
// выполняется
всегда!
эквивалентно
коду:
if(x==1)
{
   [пустой
оператор];
}
y=x+3;
```

Найдите ошибки:

```
if(x==1)
y=x+3;
z=y*5;
```

// неправильно! отсутствуют фигурные скобки, хотя в условии задумано больше одного оператора

эквивалентно коду:

```
if (x==1)
{
y=x+3;
}
z=y*5;
```

Домашнее задание:

- Выучить основные источники ошибок
- написать примеры возможных ошибок для нескольких операторов