КРОСС-ПЛАТФОРМЕННОЕ ПРОГРАММИРОВАНИЕ

ПРОЕКТИРОВАНИЕ ПО КОНТРАКТУ

Корректность ПО

```
int DoSomething(int y) {
    int x = y / 2;
    return x;
}
```

Корректность – это согласованность программных элементов с заданной спецификацией.

Формула корректности. Сильные и слабые условия

Пусть *A* – это некоторая операция, тогда **формула корректности (correctness formula)** – это выражение вида:

{P} A {Q}

Формула корректности, называемая также триадой (тройкой) Хоара, говорит следующее: любое выполнение *A*, начинающееся в состоянии, где *P* истинно, завершится и в заключительном состоянии будет истинно *Q*, где *A* обозначает операцию, *P* и *Q* – это утверждения, при этом *P* является предусловием, а *Q* – постусловием

Корректность ПО

Приведенная выше формула корректности определяет полную корректность (total correctness), которая гарантирует завершаемость операции A. Помимо этого существует понятие **частичной корректности (partial correctness)**, которое гарантирует выполнение постусловия Q только при условии завершения выполнения операции A.

Пример:

{x = 5} x = x ^ 2 {x > 0}

Утверждения

Утверждение – это логическое выражение, которое должно быть истинным на некотором участке программы и может включать сущности нашего ПО. Утверждениями могут служить обычные булевы выражения с некоторыми расширениями.

Предусловия и постусловия

Пусть предусловие определяется ключевым словом **require**, а постусловие – ключевым словом **ensure**. Тогда, если с некоторой функцией r связаны утверждения **require** *pre* и **ensure** *post*, то класс говорит своим клиентам следующее:

“Если вы обещаете вызвать *r* в состоянии, удовлетворяющем *pre*, то я обещаю в заключительном состоянии выполнить *post*”.

{Pre r} R {Post r}

Пример:

Извлечение квадратного корня

//Math.cs

```
public class Math {
  public static double Sqrt(double x) {
    Contract.Requires(x >= 0, "Positive x");
    // Реализация метода
  }
}
```

Пример:

Добавление/удаление элемента стека

//Stack.cs

```
public class Stack<T> {
  public void Push(T t) {
    Contract.Requires(t != null, "t not null");
    Contract.Ensures(Count == Contract.OldValue(Count)
  + 1,
            "One more item");
    // Реализация метода
  }
```

```
public T Pop() {
    Contract.Requires(Count != 0, "Not empty");
    Contract.Ensures(Contract.Result<T>() != null,
            "Result not null");
    Contract.Ensures(Count == Contract.OldValue(Count)
  - 1,
            "One fewer item");
    // Реализация метода
  }

  public int Count {get;}
}
```

Инварианты класса

Предусловия и постусловия характеризуют конкретные функции, но не класс в целом. Экземпляры класса часто обладают некоторым глобальными свойствами, которые должны выполняться после создания объекта и не нарушаться в течение всего времени жизни. Такие свойства носят название ***инвариантов класса*** (class invariants) и они определяют более глубокие семантические свойства и ограничения, присущие объектам этого класса.

Такими свойствами могут служить:

- ***внутренние условия*** (некоторое поле не равно null или количество элементов неотрицательно);

- **правила согласованности внутреннего состояния объекта:** (empty = (count = 0)

  deposit_list.total – withdrawals_list.total = balance).

Каждое такое условие не связывает конкретные функции, а характеризует объект в каждый устойчивый момент времени.

Корректность ПО

Утверждение *Inv* является корректным инвариантом класса, если и только если оно удовлетворяет следующим двум условиям:

1) Каждая процедура создания, применимая к аргументам, удовлетворяющим ее предусловию в состоянии, в котором атрибуты имеют значения, установленные по умолчанию, вырабатывает заключительное состояние, гарантирующее выполнение *Inv*.

2) Каждая экспортируемая процедура класса, примененная к аргументам в состоянии, удовлетворяющем *Inv* и предусловию, вырабатывает заключительное состояние, гарантирующее выполнение *Inv*.

Первое условие определяет роль функции создания в жизненном цикле объекта класса и может быть выражено формально следующим образом:

{*Default* and *pre*} *body* {*post* and *Inv*}

где *Default* – состояние объекта некоторого класса со значениями полей по умолчанию (которое зависит от конкретного языка или платформы), *pre* – предусловие, *body* – тело конструктора, *post* – постусловие, *Inv* – инвариант класса.