Свойство 1. Для любой конструкции S мы имеем $\mathbf{wp}(S, F) = F$.

«закон исключенного чуда»

Свойство 2 (монотонность)**.** Для любой конструкции S и любых постусловий Q и R, таких, что $Q \rightarrow R$ для всех состояний справедливо отношение $\mathbf{wp}(S, Q) \rightarrow \mathbf{wp}(S, R)$ для всех состояний.

При любом начальном состоянии, удовлетворяющем $\mathbf{wp}(S, Q)$, согласно определению, по окончании работы системы будет $Q=1$. Следовательно, $R=1$, и начальное состояние будет удовлетворять и условию $\mathbf{wp}(S, R)$.

Свойство 3. Для любой конструкции S и для любых постусловий Q и R справедливо $\mathbf{wp}(S, Q) \wedge \mathbf{wp}(S, R) \leftrightarrow \mathbf{wp}(S, Q \wedge R)$.

1. В любой точке пространства состояний из левой части отношения логически следует его правая часть, потому что для любого начального состояния, удовлетворяющего и $\mathbf{wp}(S, Q)$, и $\mathbf{wp}(S, R)$, в совокупности установится конечное состояние, удовлетворяющее и Q и R.

   $\mathbf{wp}(S, Q) \wedge \mathbf{wp}(S, R) \rightarrow \mathbf{wp}(S, Q \wedge R$

2. Тавтологии: $(Q \wedge R) \rightarrow Q$ и $(Q \wedge R) \rightarrow R$.

   По свойству 2: $\mathbf{wp}(S, Q \wedge R) \rightarrow \mathbf{wp}(S, Q)$, $\mathbf{wp}(S, Q \wedge R) \rightarrow \mathbf{wp}(S, R)$.

   Правило вывода: $\dfrac{A \rightarrow B,\ A \rightarrow C}{A \rightarrow (B \wedge C)}$.

   $\mathbf{wp}(S, Q \wedge R) \rightarrow \mathbf{wp}(S, Q) \wedge \mathbf{wp}(S, R)$.

Свойство 4. Для любой конструкции S и для любых постусловий Q и R справедливо $\mathbf{wp}(S, Q) \vee \mathbf{wp}\ (S, R) \rightarrow \mathbf{wp}(S, Q \vee R)$.

Тавтологии: $Q \rightarrow (Q \vee R)$ и $R \rightarrow (Q \vee R)$.

По свойству 2: $\mathbf{wp}(S, Q) \rightarrow \mathbf{wp}(S, Q \vee R)$, $\mathbf{wp}\ (S, R) \rightarrow \mathbf{wp}(S, Q \vee R)$.

Правило вывода: $\dfrac{A \rightarrow C,\ B \rightarrow C}{(A \vee B) \rightarrow C)}$.

$\mathbf{wp}(S, Q) \vee \mathbf{wp}\ (S, R) \rightarrow \mathbf{wp}(S, Q \vee R)$.

СЕМАНТИКА ЯЗЫКА ПРОГРАММИРОВАНИЯ

Оператор SKIP (ПРОПУСТИТЬ)

$\forall R$ **wp**(SKIP, R)=R

Оператор FAIL (ОТКАЗАТЬ)

$\forall R$ **wp**(SKIP, R)=F

<оператор>::=SKIP | FAIL

Оператор присваивания

x:=E

wp(x:=E, R)=$R_{E\to x}$

$|Dom(E) \wedge R_{E\to x}\}$

<оператор присваивания>::=<переменная>:=<выражение> |
<переменная>, <оператор присваивания> |
<переменная>, <оператор присваивания>, <выражение>

<оператор>::=SKIP | FAIL | <оператор присваивания>

wp(S_1, S_2, R)= **wp**(S_1, **wp**(S_2, R))

<охраняющий заголовок>::=<логическое выражение>→<оператор>
<охраняемая команда>::=<охраняющий заголовок>{;< оператор>}
<набор охраняемых команд>:: = <охраняемая команда>
<набор охраняемых команд>:: = <охраняемая команда>
{□ <охраняемая команда}
<оператор>:: = **if**<набор охраняемых команд> **fi**

1. Все предохранители определены; если вычисление какого-то предохранителя может привести к работе без правильного завершения, то и вся конструкция не сможет правильно завершить свою работу.
2. Конструкция приведет к недетерминированности при тех начальных состояниях, для которых истинны значения более чем одного предохранителя, поскольку остается неопределенным, какой из соответствующих списков операторов будет тогда выбираться для запуска. Никакой недетерминированности не возникает, если все предохранители попарно исключают друг друга.
3. Если начальное состояние таково, что ни один из предохранителей не является истиной, то это начальное состояние, для которого не подходит ни один из вариантов, а следовательно, и вся конструкция в целом. Запуск при таком начальном состоянии приведет к отказу.

Оператор «IF»

$$\textbf{if } B_1 \rightarrow SL_1 \square \ldots \square B_n \rightarrow SL_n \textbf{ fi}$$

$$\textbf{wp}(\text{IF}, R) = (\exists j: 1<j<n: B_j) \wedge (\forall j: 1<j<n: B_j \rightarrow \textbf{wp}(SL_j, R))$$

Оператор «DO»

$$\textbf{do } B_1 \rightarrow SL_1 \square \ldots \square B_n \rightarrow SL_n \textbf{ od}$$

Если «IF» – оператор, полученный путем заключения того же набора охраняемых команд в скобки «if ... fi», и условия $H_k(R)$ заданы в виде

$$H_0(R) = R \wedge \neg (\exists j: B_j),$$

$$\text{и при } k>0 \; H_k(R) = \textbf{wp}(\text{IF}, H_{k-1}(R)) \vee H_0(R),$$

$$\text{то } \textbf{wp}(\text{DO}, R) = (\exists k \geq 0: H_k(R)).$$

Интуитивно $H_k(R)$ интерпретируется как слабейшее предусловие, такое, что конструкция **do-od** завершит свою работу после не более чем k выборок охраняемых команд, причем система останется в конечном состоянии, удовлетворяющем постусловию R.

При k=0 требуется, чтобы конструкция **do-od** завершила работу, не производя выборки какой-либо охраняемой команды.
Нет ни одного предохранителя с истинным значением:

$$H_0(R)=R \wedge \neg (\exists j: B_j).$$

Начальная истинность R – необходимое условие для конечной истинности R:

$$H_0(R)=\boldsymbol{R} \wedge \neg (\exists j: B_j).$$

При k>0 есть два случая:

1. Ни один из предохранителей не имеет истинного значения, но тогда выполняется условие R, и $H_k(R)=$ **wp**(IF, $H_{k-1}(R)) \vee \boldsymbol{H_0(R)}$.

2. Хотя бы один предохранитель имеет значение истина, что соответствует однократному запуску оператора «IF» (в начальном состоянии, которое не может привести к немедленному отказу вследствие отсутствия истинных значений предохранителей).
После такого выполнения, при котором производилась выборка одной охраняемой команды, необходима гарантия попадания в состояние, при котором потребуется не более чем (k–1) дальнейших выборок для достижения завершения работы в конечном состоянии, удовлетворяющем R.
В соответствии с определением, таким постусловием для этого оператора «IF» служит $H_{k-1}(R)$.
Согласно определению **wp**(DO, R) должно существовать такое значение k, чтобы потребовалось не более чем k выборок для обеспечения завершения работы в конечном состоянии, удовлетворяющем постусловию R.

Рассмотрим конструкции, выводимые из набора охраняемых команд

$$B_1 \to S_1 \,\square\, \ldots \square\, B_n \to S_n$$

Обозначим **«IF»** и **«DO»** операторы, получаемые заключением этого набора охраняемых команд в пары скобок **«if ... fi»** и **«do ... od»** соответственно.

***Теорема* 1**. **(*Основная теорема для конструкции выбора*)** Пусть конструкция выбора **IF** и пара предикатов Q и R таковы, что

$$Q \to \exists j\ B_j \quad (1 \le j \le n) \qquad (1)$$

$$(\forall j\ B_j\ \ Q \wedge B_j \to \mathbf{wp}\,(S_j,\ R)) \qquad (2)$$

одновременно справедливы для всех состояний. Тогда

$$Q \to \mathbf{wp}\,(\mathbf{IF}, R) \qquad (3)$$

справедливо также для всех состояний.

По определению,

$$\mathbf{wp}\,(\,\mathbf{IF},\,R) = \exists\, j\; B_j \;\; \wedge (\forall\, j\; B_j\; (\,Q \wedge B_j \;\rightarrow \mathbf{wp}\,(\,S_j\,,\;\; R\,))).$$

Согласно (1), из Q следует первый член правой части, то отношение (3) доказывается, если на основании (2) мы можем сделать вывод, что

$$Q \rightarrow (\forall\, j\; B_j\; (B_j \;\rightarrow \mathbf{wp}\,(\,S_j\,,\;\; R\,))) \qquad (4)$$

справедливо для всех состояний.
Если $Q = \mathbf{F}$, отношение (4) истинно.
Если $Q = \mathbf{T}$, то для любого значения j различаются два случая:

1) $B_j = \mathbf{F}$, то $B_j \;\rightarrow \mathbf{wp}\,(\,S_j\,,\;\; R\,)$ ***истина***;

2) $B_j = \mathbf{T}$, то, согласно (2), $\mathbf{wp}\,(\,S_j\,,\;\; R\,) = \mathbf{T}$, а следовательно,

$B_j \;\rightarrow \mathbf{wp}\,(\,S_j\,,\;\; R\,)$ тоже ***истина***.

Следствие. **($n = 2$)** и $B_2 = \neg B_1$**.** Тогда $\exists j\ B_j = \mathrm{T}$, и слабейшее предусловие преобразуется так:

$$(B_1 \Rightarrow \mathrm{wp}(S_1, R)) \wedge (\neg B_1 \rightarrow \mathrm{wp}(S_2, R)) =$$

$$(\neg B_1 \vee \mathrm{wp}(S_1, R)) \wedge (B_1 \vee \mathrm{wp}(S_2, R)) =$$

$$(B_1 \wedge \mathrm{wp}(S_1, R)) \vee (\neg B_1 \wedge \mathrm{wp}(S_2, R)) \qquad (5)$$

$$B_1 \wedge \neg B_1 = \mathrm{F}$$

$$((\mathrm{wp}(S_1, R) \wedge \mathrm{wp}(S_2, R)) = \mathrm{T}) \rightarrow$$

$$(\mathrm{wp}(S_1, R) = \mathrm{T} \wedge \mathrm{wp}(S_2, R) = \mathrm{T})$$

В случае ***if – then – else*** (3)

$$(Q \wedge B_1 \rightarrow \mathrm{wp}(S_1, R)) \wedge (Q \wedge \neg B_1 \rightarrow \mathrm{wp}(S_2, R))$$

Лемма. Пусть пара предикатов Q и R может быть, записана в виде

$$R = P$$

$$Q = P \wedge \exists\, j\, B_j$$

В этом случае посылка (1) выполняется автоматически, а поскольку,

$$(\exists\, j\, B_j \ \wedge \ B_j\ ) = B_j$$

посылка (2) сводится к виду

$$(\forall\, j\, B_j\ (P \wedge B_j \ \rightarrow \mathbf{wp}\ (\ S_j\ ,\ R\ )) \qquad (6)$$

из чего можно вывести, согласно (3),

$$(P \wedge \exists\, j\, B_j\ ) \rightarrow \mathbf{wp}\ (\mathbf{IF}, R) \qquad (7)$$

для всех состояний.

***Теорема 2.* (*Основная теорема для конструкции повторения*)**
Пусть набор охраняемых команд с построенной для него конструкцией выбора **IF** и предикат ***P*** таковы, что

$$(P \wedge \exists\, j\ B_j) \Rightarrow \mathbf{wp}\ (\mathbf{IF}, P) \qquad (7)$$

справедливо для всех состояний. Тогда для соответствующей конструкции повторения **DO** можно вывести, что

$$(P \wedge \mathbf{wp}\ (\mathbf{DO}, \mathbf{T})) \Rightarrow \mathbf{wp}\ (\mathbf{DO}, P \wedge \neg(\exists\, j\ B_j)) \qquad (8)$$

для всех состояний.

Эту теорема, известна под названием **«основная теорема инвариантности для циклов»**.
Интуитивное "доказательство".

Посылка (7) утверждает, что если

1. предикат ***P*** первоначально истинен,
2. одна из охраняемых команд выбирается для выполнения,

то после ее выполнения ***P*** сохранит свою истинность.

Иначе говоря, предохранители гарантируют, что выполнение соответствующих списков операторов не нарушит истинности ***P***, если начальное значение ***P*** было истинным. Следовательно, вне зависимости от того, как часто производится выборка охраняемой команды из имеющегося набора, предикат ***P*** будет справедлив при любой новой проверке предохранителей. После завершения всей конструкции повторения, когда ни один из предохранителей не является истиной, работа закончится в конечном состоянии, удовлетворяющем $P \wedge \neg(\exists\, j\ B_j)$.

Вопрос в том, завершится ли работа правильно. Да, если условие **wp(DO, T)** справедливо и вначале; поскольку любое состояние удовлетворяет **T**, то **wp(DO, T**) по определению является слабейшим предусловием для начального состояния, такого, что запуск оператора **DO** приведет к правильно завершаемой работе.

Формальное доказательство основной теоремы для конструкции повторения основывается на формальном описании семантики этой конструкции. Выводятся следующие утверждения.
Для постусловия T.

$$H_0(T) = \neg (\exists\, j\ B_j) \qquad (9)$$

$$k>0 \quad H_k(T) = \mathbf{wp}\,(\mathbf{IF}, H_{k-1}(T)) \vee \neg (\exists\, j\ B_j) \qquad (10)$$

Для постусловия $P \wedge \neg(\exists\, j\ B_j)$.

$$H_0(P \wedge \neg(\exists\, j\ B_j)) = P \wedge \neg(\exists\, j\ B_j) \qquad (11)$$

$$k>0 \quad H_k(P \wedge \neg(\exists\, j\ B_j)) =$$
$$\mathbf{wp}\,(\mathbf{IF}, H_{k-1}(P \wedge \neg(\exists\, j\ B_j))) \vee P \wedge \neg (\exists\, j\ B_j) \qquad (12)$$

Докажем по индукции, что посылка (7) гарантирует истинность при $k \geq 0$ $(P \wedge H_k(T)) \Rightarrow H_k(P \wedge \neg(\exists\, j\ B_j))$ (13) для всех состояний.

Формальное доказательство основной теоремы для конструкции повторения основывается на формальном описании семантики этой конструкции. Выводятся следующие утверждения.
Для постусловия T.

$$H_0(T) = \neg (\exists\, j\ B_j) \qquad (9)$$

$$k>0 \quad H_k(T) = \mathbf{wp}\,(\mathbf{IF}, H_{k-1}(T)) \vee \neg (\exists\, j\ B_j) \qquad (10)$$

Для постусловия $P \wedge \neg(\exists\, j\ B_j)$.

$$H_0(P \wedge \neg(\exists\, j\ B_j)) = P \wedge \neg(\exists\, j\ B_j) \qquad (11)$$

$$k>0 \quad H_k(P \wedge \neg(\exists\, j\ B_j)) =$$
$$\mathbf{wp}\,(\mathbf{IF}, H_{k-1}(P \wedge \neg(\exists\, j\ B_j))) \vee P \wedge \neg (\exists\, j\ B_j) \qquad (12)$$

Докажем по индукции, что посылка (7) гарантирует истинность при $k \geq 0$ $(P \wedge H_k(T)) \rightarrow H_k(P \wedge \neg(\exists\, j\ B_j))$ (13) для всех состояний.

Из (9) и (11) следует справедливость (13) при $k=0$.

Пусть (13) верно при $k = K - 1$ $(K > 0)$. Докажем его справедливость при $k = K$.

$$P \wedge H_K(T) = P \wedge \mathbf{wp}(\mathbf{IF}, H_{K-1}(T)) \vee P \wedge \neg(\exists j\, B_j) =$$

в силу (10)

$$= P \wedge \exists j\, B_j \wedge \mathbf{wp}(\mathbf{IF}, H_{K-1}(T)) \vee P \wedge \neg(\exists j\, B_j) =$$

т.к. выполнилось $H_{k-1}(T)$, значит выполнилось $\exists j\, B_j$

$$= \mathbf{wp}(\mathbf{IF}, P) \wedge \mathbf{wp}(\mathbf{IF}, H_{K-1}(T)) \vee P \wedge \neg(\exists j\, B_j) =$$

в силу (7)

$$= \mathbf{wp}(\mathbf{IF}, P \wedge H_{K-1}(T)) \vee P \wedge \neg(\exists j\, B_j) =$$

в силу св. 3 преобразователя предикатов

$$= \mathbf{wp}(\mathbf{IF}, H_{K-1}(P \wedge \neg(\exists j\, B_j))) \vee P \wedge \neg(\exists j\, B_j) =$$

в силу св. 2 преобразователя предикатов и (13)

$= H_K(P \wedge \neg(\exists j\, B_j))$. (13) доказано.

$$P \wedge \mathbf{wp}(\mathbf{DO}, T) = \exists k \geq 0\; P \wedge H_k(T) \rightarrow \exists k \geq 0\; H_K(P \wedge \neg(\exists j\, B_j)) =$$

$$\mathbf{wp}(\mathbf{DO}, P \wedge \neg(\exists j\, B_j))$$

Метод индуктивных утверждений,
независимо сформулированный К. Флойдом и П. Науром.

Суть этого метода состоит в следующем:

1) формулируются входное и выходное утверждения: входное утверждение описывает все необходимые входные условия для программы (или программного фрагмента), выходное утверждение описывает ожидаемый результат;

2) предполагая истинным входное утверждение, строится промежуточное утверждение, которое выводится на основании семантики операторов, расположенных между входом и выходом (входным и выходным утверждениями); такое утверждение называется выведенным утверждением;

3) формулируется теорема (условия верификации):

из выведенного утверждения следует выходное утверждение;

4) доказывается теорема; доказательство свидетельствует о правильности программы (программного фрагмента).

Доказательство проводится при помощи математических методов, использующих исчисление предикатов первого порядка.

Введем обозначение **{ Q } S { R }**, где Q, R-предикаты, S-программа (оператор или последовательность операторов).

Обозначение определяет следующий смысл: "Если выполнение S началось в состоянии, удовлетворяющем Q, то имеется гарантия, что оно завершится через конечное время в состоянии, удовлетворяющем R".

Предикат Q называется *предусловием* или входным утверждением S, предикат R - *постусловием* или выходным утверждением. Следовательно, R определяет то, что нужно установить. Можно сказать, что R определяет спецификацию задачи.

A1. Аксиома присваивания: $\{ R_o \}\ x := e\ \{ R \}$

Неформальное объяснение аксиомы: так как x после выполнения будет содержать значение e, то R будет истинно после выполнения, если результат подстановки e вместо x в R истинен перед выполнением.

Таким образом $R_o = R(x)$ при $x = e$.

Для R_o вводится обозначение: $R_o = R^x_e$ (у Вирта) или

$R_{x \leftarrow e}$ (у Дейкстры)

что означает, что x заменяется на e.

Аксиома присваивания будет иметь вид:

$\{ R_{x \leftarrow e} \}\ x := e\ \{R\}$

A2. Если известно: { Q } S { P } и { P } $\Rightarrow$ { R }, то
{ Q } S { R }

A3. Если известно: { Q } S { P } и { R } $\Rightarrow$ { Q }, то
{ R } S { P }

Пусть S – это последовательность из двух операторов S1;S2 (составной оператор).

A4. Если известно:
{ Q } S_1 { P1 } и { P1 } S_2 { R }, то { Q } S { R }.

A5. Если известно:
{ Q and B } S_1 { R } и { Q not B } $\Rightarrow$ { R }, то
{ Q } if B then S_1 { R }.

A6. Если известно:
{ Q and B } S_1 { R } и { Q not B } S_2 { R }, то
{ Q } if B then S_1 else S_2 { R }

A7. Если известно: { Q and not B } S_1 { Q } , то
{ Q } repeat S_1 until B { Q and B }

A8. Если известно: { Q and B } S1 { Q } , то
{ Q } while B do S_1 { Q and not B }

Предикат Q, истинный перед выполнением и после выполнения каждого шага цикла, называется *инвариантным отношением* или просто *инвариантом цикла.*

В математике термин "инвариантный" означает не изменяющийся под воздействием совокупности рассматриваемых математических операций. В данном случае единственная операция – это выполнение шага цикла при условии истинности Q вначале.

Пусть надо определить частное q и остаток r от деления x на y.

Входные данные:

x, y принадлежат классу целых неотрицательных чисел, причем y > 0.

Выходные данные:

q, r принадлежат классу целых неотрицательных чисел.

Описание алгоритма

```
Задать(x,y); /* x,y получают конкретные значения X,Y */
r := x;
q := 0;
while y <= r do
begin
r := r - y; q := q + 1
end;
выдать(q,r);
```

Сформулируем постусловие R:

$(r < y)$ and $(x = y*q + r)$

Нужно доказать, что

$\{ y > 0$ and $x > y \}$ S $\{ (r < y)$ and $(x = y*q + r) \}$

Доказательство.

1. Очевидно, что Q => x = x + y * 0.
2. Применим аксиому A1 к оператору r := x, тогда получим
{ x = x + y * 0 } r := x { x = r + y * 0 }
3. Аналогично, применяя A1 к оператору q := 0, получим:
{ x = r + y * 0 } q := 0 { x = r + y*q }
4. Применяя правило A3 к результатам пунктов 1 и 2, получим
{ Q } r := x { x = r + y * 0 }
5. Применяя правило A4 к результатам пунктов 4 и 3, получим
{ Q } r := x; q := 0 { x = r + y*q }
6. Выполним равносильное преобразование
x = r + y * q and y< =r= > x = (r - y) + y*(q + 1)

7. Применяя правило A1 к оператору r := r - y, получим
{x = (r - y) + y*(q + 1)} r := r - y {x = r + y*(q + 1)}
8. Для оператора q := q + 1 аналогично получим
{ x = r + y*(q + 1) } q := q + 1 { x = r + y*q }
9. Применяя правило A4 к результатам пунктов 7 и 8, получим
{ x = (r - y) + y*(q + 1) }
r := r - y; q := q + 1
{ x = r + y*q }
10. Применяя правило A2 к результатам пунктов 6 и 9, получим
{ x = r + y*q and y <= r }
r := r - y; q := q + 1
{ x = r + y*q }
11. Применяя правило A8 к результату пункта 10, получим
{ x = r + y*q } while y <= r do
beqin r := r - y; q := q + 1 end
{ not (y <= r) and (x = r + y*q) }
Утверждение x = r + y*q является инвариантом цикла, так как значение его остается истинным до цикла и после выполнения каждого шага цикла.

12. Применяя правило A4 к результатам пунктов 5 и 11, получаем то, что требовалось доказать, т.е.

{ Q } S { not (y <= r) and (x = r + y*q) }

Осталось доказать, что выполнение программы заканчивается. Доказывать будем от противного, т.е. предположим, что программа не заканчивается. Тогда должна существовать бесконечная последовательность значений r и q, удовлетворяющая условиям

1) y <= r ;

2) r, q принадлежат классу неотрицательных целых чисел.

Но значение r на каждом шаге цикла уменьшается на положительную величину:

r := r - y (y > 0).

Значит, последовательность значений r и q является конечной, т.е. найдется такое значение r, для которого не будет выполняться условие y <= r и циклический процесс завершится.