

12

Синтезиране на програми на подмножество на езика C++

12.1 Необходимост от синтезиране на програми

Нека като част от алголоподобна програма се налага да се намерят частното y_1 и остатъкът y_2 от целочисленото деление на неотрицателното цяло число x_1 на положителното цяло число x_2 . За решение нека сме написали следния програмен фрагмент:

```
y1=0; y2=x1;
while (y2>x2)
{y1++;
 y2=y2-x2;
}
```

Трябва да проверим дали той наистина решава задачата. За целта най-често се извършва тестване, при което за допустими входни данни се проверява дали се получава правилен резултат. От условието се вижда, че за входни стойности трябва да се изберат такива, които удовлетворяват: $x_1 \geq 0 \wedge x_2 > 0$, където x_1 и x_2 са цели числа, а \wedge означава конюнкция. Когато изпълнението на програмния фрагмент завърши, трябва да е в сила: $x_1 = y_1 \cdot x_2 + y_2 \wedge 0 \leq y_2 < x_2$. Условието $y_2 < x_2$ следва от дефиницията за остатък от целочислено деление. Свързваме програмния фрагмент с подходящи коментари, означаващи тези условия, т.е.

```
{φ(x1,x2): x1≥0 ∧ x2>0}
```

```

y1=0; y2=x1;
while(y2>x2)
{y1++;
  y2=y2-x2;
}
{ψ(x1,x2,y1,y2): x1=y1.x2+y2 ∧ 0≤y2<x2}

```

Предикатът $\varphi(x1,x2)$ се нарича **предусловие**, а $\psi(x1,x2,y1,y2)$ - **следусловие**. За да извършим тестването, ще оградим програмния фрагмент със следните оператори:

```

cout << "делимо x1= " << x1
      << " делител x2= " << x2 << endl;

```

- пред фрагмента

и

```

cout << "x1 = y1.x2+y2 " << (x1 == y1*x2+y2) << endl
      << " 0 <= y2 < x2 = " << (0 <= y2 && y2 < x2) << endl;

```

- след фрагмента.

При изпълнение на програмния фрагмент за $x1==7$ и $x2==2$, се получава:

```

делимо x1 = 7 делител x2 = 2
x1 = y1.x2+y2 = 1
0 <= y2 < x2 = 1,

```

а при изпълнение за $x1 == 9$ и $x2 == 3$ -

```

делимо x1 = 9 делител x2 = 3
x1 = y1.x2+y2 = 1
0 <= y2 < x2 = 0.

```

Резултатът не е правилен, тъй като условието $y2<x2$ не е в сила ($y1==2$ и $y2==3$).

След постъпкови изпълнения заключаваме, че ако заменим условието $y2>x2$ на оператора за цикъл с $y2>=x2$, изпълнението на програмния фрагмент

```

{φ(x1,x2): x1≥0 ∧ x2>0}
y1=0; y2=x1;
while (y2>=x2)
{y1++;
  y2=y2-x2;
}

```

$$\{\psi(x_1, x_2, y_1, y_2): x_1 = y_1 \cdot x_2 + y_2 \wedge 0 \leq y_2 < x_2\}$$

ще даде правилни резултати в горните два случая. Отново не сме сигурни, дали за всички цели числа x_1 и x_2 , които удовлетворяват предиката $\varphi(x_1, x_2)$, след завършване на изпълнението на програмния фрагмент, ще е в сила $\psi(x_1, x_2, y_1, y_2)$ за текущите стойности на x_1 , x_2 , y_1 и y_2 . Ще отбележим също, че особено важно е правилното определяне на предусловието φ и следусловието ψ .

Но даже и правилно да сме ги определили, след завършване на тестването на програмния фрагмент отново не можем да сме съвсем сигурни, че той е правилен. Можем само да заключим, че фрагментът дава правилни резултати за тези частни случаи, които са използвани при тестването.

За да се убедим, че програма (програмен фрагмент) е правилна относно дадена входно/изходна спецификация, най-добре е с формални средства да извършим доказателство. Такива средства са методите за верификация на програми.

През последните години, доказателството на правилността на програмите заема сравнително важно място в науката информатика. Процесът на програмиране се състои в писане на програми, аотиране на програмите с предусловия и следусловия, удовлетворяващи дадена входно/изходна спецификация и доказване на правилността им чрез различни методи за верификация. Всички популярни методи за верификация на програми обаче са трудоемки. Това води до необходимостта програмите да бъдат синтезирани, т.е. да бъдат строени паралелно с доказателството, че са правилни.

Съществуват различни методи за синтезиране на алголоподобни програми. В настоящата глава ще илюстрираме подхода на преобразуващите предикати, който може да се използва както за синтезиране, така и за верифициране на алголоподобни програми, в частност на програми на езика C++.

12.2 Синтезиране на програми на подмножество на C++

За да не усложняваме изложението, ще използваме подмножество на езика, състоящо се от: празен оператор, блок, оператор за присвояване, условен оператор (пълна и кратка форма) и оператор за цикъл `while`.

За синтезирането на програми ще използваме специален предикат, който се нарича **преобразуващ предикат**.

Дефиниция 1. Нека S е произволен оператор, а R е предикат, който описва очаквания от изпълнението на оператора S резултат. **Преобразуващ предикат** за S и R е предикатът $wp(S, R)$, представящ множеството от всички състояния, за които изпълнението на S , започващо от такова състояние, завършва и е в сила изходният предикат R .

Примери:

а) Нека S е оператора за присвояване $a=a+3$, а R е предиката $a \leq 3$. Тогава $wp(a=a+3, a \leq 3) = (a \leq 0)$, тъй като ако $a \leq 0$, изпълнението на $a=a+3$ завършва и след завършването е в сила $a \leq 3$. Освен това, ако е в сила $a > 0$, изпълнението на оператора за присвояване $a=a+3$ завършва, но не може да направи да е в сила $a \leq 3$.

б) Нека S е оператора `if (a>=b)c=b; else c=a;` а R е предиката $c=\min(a,b)$. Изпълнението на оператора S винаги присвоява на c стойността на $\min(a,b)$. Следователно $wp(S,R)=T$, където с T е означена константата `true`.

Отначало ще определим преобразувания предикат за операторите на едно подмножество на езика.

Нека R е произволен предикат.

Дефиниция 2. $wp(\text{празен_оператор}, R) = R$.

Дефиниция 3. $wp(x = e, R) = \text{domain}(e) \wedge R(x \leftarrow e)$, където $\text{domain}(e)$ е предикат, описващ множеството от всички състояния, в които е дефиниран изразът e , а $R(x \leftarrow e)$ е предикат, който се получава като в R променливата x се замени с e .

Пример: $wp(x=x-2, x=10) = (x-2=10) = (x=12)$

Дефиниция 4. $wp(S1;S2;\dots;Sn, R) = wp(S1, wp(S2;\dots;Sn, R))$, където $S1, S2, \dots, Sn$ са произволни оператори.

Пример:
$$\begin{aligned} & \text{Wp}(t=x; x=y; y=t, x=x_0 \wedge y=y_0) = \\ & \text{Wp}(t=x, \text{Wp}(x=y; y=t, x=x_0 \wedge y=y_0)) = \\ & \text{Wp}(t=x, \text{Wp}(x=y, \text{Wp}(y=t, x=x_0 \wedge y=y_0))) = \\ & \text{Wp}(t=x, \text{Wp}(x=y, x=x_0 \wedge t=y_0)) = \\ & \text{Wp}(t=x, y=x_0 \wedge t=y_0) = (y=x_0) \wedge (x=y_0), \end{aligned}$$

където x_0 и y_0 са произволни начални стойности за променливите x и y .

Дефиниция 5. $\text{Wp}(\{S_1; S_2; \dots; S_n\}, R) = \text{Wp}(S_1; S_2; \dots; S_n, R)$,
където S_1, S_2, \dots, S_n са произволни оператори.

Дефиниция 6.
$$\begin{aligned} \text{Wp}(\text{if}(B)S_1; \text{else } S_2, R) = & \text{domain}(B) \wedge \\ & (B \Rightarrow \text{Wp}(S_1, R)) \wedge \\ & (\neg B \Rightarrow \text{Wp}(S_2, R)). \end{aligned}$$

В частност

$$\text{Wp}(\text{if}(B)S, R) = \text{domain}(B) \wedge (B \Rightarrow \text{Wp}(S, R)) \wedge (\neg B \Rightarrow R).$$

Често не е необходимо да се намери преобразуващият предикат $\text{Wp}(\text{if}(B)S_1; \text{else } S_2, R)$ или $\text{Wp}(\text{if}(B)S, R)$, а само да се провери дали е в сила импликацията $Q \Rightarrow \text{Wp}(\text{if}(B)S_1; \text{else } S_2, R)$ или $Q \Rightarrow \text{Wp}(\text{if}(B)S, R)$. В тези случаи е полезна следната теорема.

Теорема 1. Нека за оператора
 $\text{if}(B)S_1; \text{else } S_2$

и предикатите Q и R са в сила условията:

- а) $Q \Rightarrow \text{domain}(B)$
- б) $Q \wedge B \Rightarrow \text{Wp}(S_1, R)$
- в) $Q \wedge \neg B \Rightarrow \text{Wp}(S_2, R)$.

Тогава (и само тогава) е в сила

$$Q \Rightarrow \text{Wp}(\text{if}(B)S_1; \text{else } S_2, R).$$

Доказателството на тази теорема е дадено в [4].

Следствие. Нека за оператора
 $\text{if}(B)S;$

и предикатите Q и R са в сила условията:

- а) $Q \Rightarrow \text{domain}(B)$
- б) $Q \wedge B \Rightarrow \text{Wp}(S, R)$
- в) $Q \wedge \neg B \Rightarrow R$.

Тогава (и само тогава) е в сила $Q \Rightarrow wp(\text{if}(B) S, R)$.

Забележка. Ако операторът $\text{if}(B) S_1; \text{else } S_2;$ или $\text{if}(B)S;$ се изпълнява в състояние, при което предикатът B е дефиниран, т.е. $\text{domain}(B) = T$, условие а) от Теорема 1 и следствието отпада.

От Теорема 1 и следствието от нея, ще определим неформално метода на преобразуващите предикати за синтезиране на условен оператор.

Метод на преобразуващите предикати за

а) синтезиране на оператора *if/else*

1. Намират се условие B , оператори S_1 и S_2 , така че импликациите:

$$Q \wedge B \Rightarrow wp(S_1, R) \text{ и}$$

$$Q \wedge \neg B \Rightarrow wp(S_2, R) \text{ да са в сила.}$$

2. Проверява се дали е в сила $Q \Rightarrow \text{domain}(B)$.

б) синтезиране на оператора *if* – кратка форма

1. Намират се условие B и оператор S , така че импликациите:

$$Q \wedge B \Rightarrow wp(S, R) \text{ и}$$

$$Q \wedge \neg B \Rightarrow R \text{ да са в сила.}$$

2. Проверява се дали е в сила $Q \Rightarrow \text{domain}(B)$.

Тъй като практически не е лесно да се използва дефиницията на преобразуващия предикат за оператора while , ще формулираме теорема, чрез която може да се провери верността на импликацията

$$Q \Rightarrow wp(\text{while}(B)S, R).$$

За целта с оператора за цикъл $\text{while}(B)S;$ свързваме:

а) инварианта P – предикат, който е в сила преди изпълнението и след изпълнението на всяка стъпка на оператора while ;

б) ограничаваща функция t – целочислена функция, явяваща се горна граница на броя на стъпките на цикъла, които остават да бъдат изпълнени. Функцията t трябва да е ограничена отдолу от 0 и при всяка стъпка от изпълнението на оператора за цикъл да намалява поне с 1.

Теорема 2. Нека предикатът P и целочислената функция t удовлетворяват условията:

$$a) P \wedge B \Rightarrow wp(S, P)$$

$$b) P \wedge B \Rightarrow t > 0 \text{ и}$$

$$в) P \wedge B \Rightarrow wp(t1 = t; S, t < t1),$$

където $t1$ е нов идентификатор. Тогава е в сила условието:

$$P \Rightarrow wp(\text{while}(B)S, P \wedge \neg B).$$

Доказателството на теоремата е дадено в [4].

Като следствие от Теорема 2 може да се формулира списък от условия за верификация и синтез на оператора за цикъл `while`.

Нека е даден `while` цикъл, подходящо аотиран с предусловие, инварианта, ограничаваща функция и следусловие:

{Q: предусловие}

{P: инварианта}

{t: ограничаваща функция}

`while (B) S;`

{R: следусловие}

Списък от условия за верификация и синтез на оператора за цикъл `while`

1) P е в сила преди изпълнението на оператора за цикъл, т.е. $Q \Rightarrow P$ е в сила, или съществува оператор S_0 , така че $Q \Rightarrow wp(S_0, P)$ е в сила.

2) $P \wedge B \Rightarrow wp(S, P)$ е в сила, т.е. P е инварианта на цикъла.

3) $P \wedge \neg B \Rightarrow R$ е вярно, т.е. в момента на завършване на изпълнението на оператора за цикъл е в сила следусловието.

4) $P \wedge B \Rightarrow t > 0$ е в сила, т.е. t е ограничена отдолу от 0, докато изпълнението на оператора за цикъл не е завършило.

5) $P \wedge B \Rightarrow wp(t1 = t; S, t < t1)$ е вярно, т.е. всяка стъпка от изпълнението на оператора за цикъл води до строго намаляване на ограничаващата функция t .

На базата на този списък ще опишем метод за синтезиране на програмни фрагменти, съдържащи оператора за цикъл `while`.

Метод на преобразуващите предикати за синтезиране

на оператора за цикъл while

1. Проверява се дали е в сила $Q \Rightarrow P$. Ако това не е така, търси се оператор S_0 , така че $Q \Rightarrow wp(S_0, P)$ да е в сила.
2. Намира се условие B , така че $P \wedge \neg B \Rightarrow R$ да е в сила.
3. Проверява се дали $P \wedge B \Rightarrow t > 0$ е в сила.
4. Намира се оператор S , така че да са в сила условията:
 $P \wedge B \Rightarrow wp(t1 = t; S, t < t1)$ и
 $P \wedge B \Rightarrow wp(S, P)$.

Чрез примери ще илюстрираме метода на преобразуващите предикати за синтезиране на програми на езика C++.

Задача 104. да се синтезира програма на езика C++, която намира частното y_1 и остатъка y_2 от целочисленото деление на неотрицателното цяло число x_1 на положителното цяло число x_2 .

От условието на задачата, за предусловие и следусловие определяме:

$$Q: x_1 \geq 0 \wedge x_2 > 0$$

$$R: x_1 = y_1 \cdot x_2 + y_2 \wedge 0 \leq y_2 < x_2.$$

Тук y_1 и y_2 са цели числа, съдържащи съответно частното и остатъка от целочисленото деление x_1 на x_2 .

Тъй като y_1 е равно на броя на срещанията на x_2 в x_1 , търсената програма ще съдържа оператор за цикъл. За целта определяме следните инварианта и ограничаваща функция:

$$P: x_1 = y_1 \cdot x_2 + y_2 \wedge 0 \leq y_2 \wedge x_2 > 0$$

$$t: y_2.$$

Инвариантата е получена като от R е отстранен конюнктивният член $y_2 < x_2$ и е добавен $x_2 > 0$, съдържащ се в предусловието. Тъй като не е в сила $Q \Rightarrow P$, търсим оператор (редица от оператори) S_0 , така че $Q \Rightarrow wp(S_0, P)$ да е в сила. Нека

$$S_0: y_1 = f_1(x_1, x_2); y_2 = f_2(x_1, x_2, y_1).$$

Имаме:

$$Q \Rightarrow wp(y_1 = f_1(x_1, x_2); y_2 = f_2(x_1, x_2, y_1), P), \text{ т.е.}$$

$$x_1 \geq 0 \wedge x_2 > 0 \Rightarrow$$

$$x_1 = f_1(x_1, x_2) \cdot x_2 + f_2(x_1, x_2, f_1(x_1, x_2)) \wedge$$

$$0 \leq f_2(x_1, x_2, f_1(x_1, x_2)) \wedge x_2 > 0$$

Ако за $f_1(x_1, x_2)$ и $f_2(x_1, x_2, y_1)$ изберем 0 и x_1 съответно, импликацията ще бъде в сила. Така получаваме S_0 : $y_1 = 0$; $y_2 = x_1$.

Тъй като P и R са известни, от условие 3, от списъка от условия за верификация и синтез на оператора `while`, намираме V . Имаме:

$$x_1 = y_1 \cdot x_2 + y_2 \wedge 0 \leq y_2 \wedge x_2 > 0 \wedge \neg V \Rightarrow$$

$$x_1 = y_1 \cdot x_2 + y_2 \wedge 0 \leq y_2 < x_2,$$

т.е. V : $y_2 \geq x_2$.

Условие 4: $P \wedge V \Rightarrow t > 0$ е в сила, тъй като $y_2 \geq x_2$ и $x_2 > 0$ са в сила.

Остава да намерим тялото на цикъла. Търсим го от вида:

$$S: y_1 = g_1(x_1, x_2, y_1, y_2);$$

$$y_2 = g_2(x_1, x_2, y_1, y_2).$$

Функциите $g_1(x_1, x_2, y_1, y_2)$ и $g_2(x_1, x_2, y_1, y_2)$ трябва да са такива, че условия 2 и 5 от списъка от условия за верификация и синтез на оператора `while`, да са в сила.

От условията:

$$P \wedge V \Rightarrow wp(t_1 = t; S, t < t_1), \text{ т.е.}$$

$$x_1 = y_1 \cdot x_2 + y_2 \wedge 0 \leq y_2 \wedge x_2 > 0 \wedge y_2 \geq x_2 \Rightarrow$$

$$g_2(x_1, x_2, g_1(x_1, x_2, y_1, y_2), y_2) < y_2,$$

и

$$P \wedge V \Rightarrow wp(S, P), \text{ т.е.}$$

$$x_1 = y_1 \cdot x_2 + y_2 \wedge 0 \leq y_2 \wedge x_2 > 0 \wedge y_2 \geq x_2 \Rightarrow$$

$$x_1 = g_1(x_1, x_2, y_1, y_2) \cdot x_2 + g_2(x_1, x_2, g_1(x_1, x_2, y_1, y_2), y_2) \wedge$$

$$0 \leq g_2(x_1, x_2, g_1(x_1, x_2, y_1, y_2), y_2) \wedge x_2 > 0,$$

определяме функциите g_1 и g_2 . Ако за $g_2(x_1, x_2, y_1, y_2)$ изберем $y_2 - x_2$, $g_2(x_1, x_2, g_1(x_1, x_2, y_1, y_2), y_2) \geq 0$ ще е в сила. Освен това $g_2(x_1, x_2, g_1(x_1, x_2, y_1, y_2), y_2) < y_2$ също е в сила, тъй като $x_2 > 0$. Условието $x_1 = x_2 \cdot g_1(x_1, x_2, y_1, y_2) + g_2(x_1, x_2, g_1(x_1, x_2, y_1, y_2), y_2)$ има вида: $x_1 = x_2 \cdot g_1(x_1, x_2, y_1, y_2) + y_2 - x_2$. Тъй като $x_1 = x_2 \cdot y_1 + y_2$ е в сила, за

$g_1(x_1, x_2, y_1, y_2)$ можем да изберем y_1+1 . Тялото на оператора за цикъл има вида: $S: y_1++; y_2=y_2-x_2$, а синтезираната програма:

```
y1=0; y2=x1;
while (y2>=x2)
{y1++;
y2=y2-x2;
}
```

Съществуват различни подходи за построяване на инварианта на цикъл. Най-често използвани са: *отстраняване на конюнктивен член, замяна на константа с променлива, комбиниране на предусловие и следусловие*. Ще ги илюстрираме чрез следващите задачи.

Задача 105. Дадено е цялото число n , $n \geq 0$. Да се синтезира програма на езика C++, която намира най-голямото цяло число a , чийто квадрат е не по-голям от n .

От условието на задачата, за предусловие и следусловие определяме:

Q: $n \geq 0$

R: $a \geq 0 \wedge a^2 \leq n \wedge n < (a+1)^2$.

I начин:

Ако отстраним третия конюнктивен член на R, получаваме инвариантата:

P: $0 \leq a \wedge a^2 \leq n$,

а за ограничаваща функция избираме:

t: $n - a^2$.

От условието "P е в сила преди изпълнението на оператора за цикъл" следва, че трябва да се намери инициализация S_0 , така че да е в сила

$Q \Rightarrow \text{wp}(S_0, P)$. Търсим S_0 от вида:

$a = f_1(n)$;

Имаме:

$n \geq 0 \Rightarrow 0 \leq f_1(n) \wedge f_1(n)^2 \leq n$.

За да е в сила тази импликация, за $f_1(n)$ избираме 0. Получаваме

$S_0: a=0$;

От условието $P \wedge \neg B \Rightarrow R$ ще намерим B. Имаме:

$$0 \leq a \wedge a^2 \leq n \wedge \neg B \Rightarrow$$

$$0 \leq a \wedge a^2 \leq n \wedge n < (a+1)^2.$$

Следователно $\neg B$: $n < (a+1)^2$, а B : $n \geq (a+1)^2$. Забелязваме, че в този случай $\neg B$ е отстранения конюнктивен член.

Условие 4 има вида:

$$0 \leq a \wedge a^2 \leq n \wedge n \geq (a+1)^2 \Rightarrow n - a^2 > 0$$

и следва от $a^2 < (a+1)^2 \leq n$. Остава да намерим тялото S на оператора за цикъл, така че да са в сила условия 2 и 5 от списъка от условия за верификация и синтез на оператора `while`. S търсим от вида:

$$a = f_2(n, a);$$

Тъй като

$$\text{wp}(t1=t; S, t < t1) = a^2 < f_2^2(n, a),$$

за да е в сила условие 5, за $f_2(n, a)$ избираме $a+h$, където $h > 0$.

Условие 2 има вида:

$$0 \leq a \wedge a^2 \leq n \wedge n \geq (a+1)^2 \Rightarrow$$

$$0 \leq a+h \wedge (a+h)^2 \leq n.$$

Ако за стойност на h изберем 1, това условие ще бъде в сила и синтезираната програма има вида:

```
a=0;
while (n >= (a+1)*(a+1)) a = a+1;
```

II начин:

Ако отстраним втория конюнктивен член на R , получаваме инвариантата:

$$P: 0 \leq a \wedge n < (a+1)^2,$$

а за ограничаваща функция избираме:

$$t: (a+1)^2 - n$$

Търсим инициализация

$$S_0: a = f_1(n);$$

така че да е в сила $Q \Rightarrow \text{wp}(S_0, P)$. Имаме:

$$n \geq 0 \Rightarrow 0 \leq f_1(n) \wedge n < (f_1(n)+1)^2.$$

Ако за $f_1(n)$ изберем n , горната импликация ще бъде в сила. От условието $P \wedge \neg B \Rightarrow R$, т.е.

$$0 \leq a \wedge n < (a+1)^2 \wedge \neg B \Rightarrow$$

$$0 \leq a \wedge a^2 \leq n \wedge n < (a+1)^2,$$

получаваме $\neg B$: $a^2 \leq n$ и B : $n < a^2$. Условие 4 има вида:

$$0 \leq a \wedge n < (a+1)^2 \wedge n < a^2 \Rightarrow$$

$$(a+1)^2 - n > 0$$

и очевидно е в сила. Остава да се намери тялото S на оператора за цикъл така, че да са в сила условия 2 и 5 от списъка от условия за верификация и синтез на оператора за цикъл. S търсим от вида:

$$a = f_2(n, a);$$

Тъй като

$$wp(t_1 = t; S, t < t_1) = (f_2(n, a) + 1)^2 < (a+1)^2,$$

за $f_2(n, a)$ можем да изберем $a-h$, където $h > 0$. При този избор, условие 2 има вида:

$$0 \leq a \wedge n < (a+1)^2 \wedge n < a^2 \Rightarrow$$

$$0 \leq a-h \wedge n < (a-h+1)^2.$$

Ако за h изберем 1, тъй като $n \geq 0$, това условие ще бъде в сила и синтезираната програма има вида:

```
a=n;
while (n<a*a) a--;
```

III начин:

Инвариантата P намираме, като заменим $a+1$ от R с променливата b и укажем границите на изменение на b , т.е.

$$P: a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1,$$

a за ограничаваща функция избираме:

$$t: b-a-1.$$

Търсим инициализация

$$S_0: a = f_1(n); b = f_2(n, a);$$

така че да е в сила импликацията $Q \Rightarrow wp(S_0, P)$, т.е.

$$n \geq 0 \Rightarrow f_1(n) \geq 0 \wedge f_1(n)^2 \leq n < f_2(n, f_1(n))^2 \wedge f_1(n) < f_2(n, f_1(n)) \leq n+1.$$

За $f_1(n)$ и $f_2(n, a)$ избираме 0 и $n+1$, съответно. Получаваме:

$$S_0: a=0; b=n+1;$$

От условието $P \wedge \neg B \Rightarrow R$, т.е.

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge \neg B \Rightarrow$$

$$a^2 \leq n < (a+1)^2$$

следва, $V: b \neq a+1$. Условието $P \wedge V \Rightarrow t > 0$, т.е.

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \Rightarrow b-a-1 > 0$$

е в сила, тъй като от $a < b$ следва $b-a-1 \geq 0$, а освен това $b \neq a+1$. Тялото на цикъла търсим от вида:

$$S: a = g1(n, a, b); b = g2(n, a, b).$$

Тъй като

$$\text{wp}(t1 = b-a-1; S, b-a-1 < t1) = \\ g2(n, g1(n, a, b), b) - g1(n, a, b) - 1 < b-a-1,$$

условие 5 има вида:

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \Rightarrow \\ g2(n, g1(n, a, b), b) - g1(n, a, b) < b-a$$

и някои възможности за да е в сила са:

- . $g1(n, a, b) = a+1; g2(n, a, b) = b$ и следователно
S1: $a++$;
- . $g1(n, a, b) = a; g2(n, a, b) = b-1$ и следователно
S2: $b--$;
- . $g1(n, a, b) = (a+b)/2; g2(n, a, b) = b$ и следователно
S3: $a = (a+b)/2$;
- . $g1(n, a, b) = a; g2(n, a, b) = (a+b)/2$ и следователно
S4: $b = (a+b)/2$;

където делението е целочислено. Тъй като $a < b$ и $b \neq a+1$ са в сила, в сила са и $a < (a+b)/2 < b$.

За оператора S1, условие 2 има вида:

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \Rightarrow \\ a+1 \geq 0 \wedge (a+1)^2 \leq n < b^2 \wedge a+1 < b \leq n+1.$$

То не е вярно, но

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \wedge (a+1)^2 \leq n \Rightarrow \\ a+1 \geq 0 \wedge (a+1)^2 \leq n < b^2 \wedge a+1 < b \leq n+1,$$

т.е.

$$P \wedge V \wedge (a+1)^2 \leq n \Rightarrow \text{wp}(a++, P)$$

е в сила, тъй като от $a < b$ следва $a \leq b-1$. Но $b-1 \neq a$ и следователно $a < b-1$.

Аналогично, за оператора S2, условие 2 има вида:

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \Rightarrow$$

$$a \geq 0 \wedge a^2 \leq n < (b-1)^2 \wedge a < b-1 \leq n+1.$$

То също не е вярно, но

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \wedge (a+1)^2 > n \Rightarrow$$

$$a \geq 0 \wedge a^2 \leq n < (b-1)^2 \wedge a < b-1 \leq n+1,$$

т.е.

$$P \wedge B \wedge (a+1)^2 > n \Rightarrow \text{wp}(b--, P)$$

е в сила, тъй като от $a < b$ и $b \neq a+1$ следва $a < b-1$, т.е. $a+1 \leq b-1$. Оттук и от $n < (a+1)^2$ следва $n < (b-1)^2$.

Получихме:

$$P \wedge B \wedge (a+1)^2 \leq n \Rightarrow \text{wp}(a++, P)$$

$$P \wedge B \wedge (a+1)^2 > n \Rightarrow \text{wp}(b--, P)$$

От Теорема 1 следва, че е в сила импликацията:

$$P \wedge B \Rightarrow \text{wp}(S, P),$$

където

S: if ((a+1)*(a+1) <= n) a++; else b--;

Така синтезираната програма има вида:

```
a=0; b=n+1;
while (b!=a+1)
  if ((a+1)*(a+1)<=n) a++;
  else b--;
```

За оператора S2, условие 2 ще е в сила също, ако вместо предиката $(a+1)^2 > n$ в лявата страна на импликацията се добави конюнктивният член $n < (b-1)^2$, т.е.

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \wedge n < (b-1)^2 \Rightarrow$$

$$a \geq 0 \wedge a^2 \leq n < (b-1)^2 \wedge a < b-1 \leq n+1$$

е в сила.

Освен това, тъй като и импликацията:

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \wedge (b-1)^2 \leq n \Rightarrow$$

$$a+1 \geq 0 \wedge (a+1)^2 \leq n < b^2 \wedge a+1 < b \leq n+1,$$

е в сила заради $(a+1)^2 \leq (b-1)^2 \leq n$, от Теорема 1 следва, че е в сила импликацията:

$$P \wedge V \Rightarrow \text{wp}(S, P),$$

където

S: if(n<(b-1)*(b-1))b--; else a ++;

Така синтезирахме още една програма, която удовлетворява дадената входно/изходна спецификация:

```
a=0; b=n+1;
while (b!=a+1)
if (n<(b-1)*(b-1)) b--;
else a++;
```

За оператора S3, условие 2 има вида:

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \Rightarrow \\ (a+b)/2 \geq 0 \wedge ((a+b)/2)^2 \leq n < b^2 \wedge (a+b)/2 < b \leq n+1$$

и не е вярно. В сила е обаче импликацията:

$$P \wedge V \wedge ((a+b)/2)^2 \leq n \Rightarrow \text{wp}(S3, P).$$

За оператора S4, условие 2 има вида:

$$a \geq 0 \wedge a^2 \leq n < b^2 \wedge a < b \leq n+1 \wedge b \neq a+1 \Rightarrow \\ a \geq 0 \wedge a^2 \leq n < ((a+b)/2)^2 \wedge a < (a+b)/2 \leq n+1$$

и също не е вярно, но е в сила:

$$P \wedge V \wedge n < ((a+b)/2)^2 \Rightarrow \text{wp}(S4, P).$$

Като използваме отново Теорема 1, за тяло на цикъла получаваме:

S: if ((a+b)/2*(a+b)/2<=n) a=(a+b)/2;
else b=(a+b)/2;

а синтезираната програма има вида:

```
a=0; b=n+1;
while (a+1!=b)
if (((a+b)/2)*((a+b)/2)<=n) a=(a+b)/2;
else b=(a+b)/2;
```

която може да се опрости до:

```
a=0; b=n+1;
while (a+1!=b)
{int x=(a+b)/2;
```

```

if (x*x<=n) a=x;
else b=x;
}

```

Получихме пет програми, които удовлетворяват дадената входно/изходна спецификация. Те са еквивалентни, тъй като са “извлечени” от една и съща входно/изходна спецификация.

Задача 106. да се синтезира програма, която по дадено естествено число n , $n \geq 1$, намира n - тото число на Фибоначи.

Анализирайки условието на задачата, за предусловие и следусловие избираме:

Q: $n \geq 1$

R: $a = \text{fib}(n)$,

където a е n -тото число на Фибоначи.

Като използваме дефиницията на числата на Фибоначи, за инвариантата и ограничаваща функция определяме:

P: $a = \text{fib}(i) \wedge b = \text{fib}(i-1) \wedge 1 \leq i \leq n$

t: $n-i$.

Изборът на P означава, че в произволен момент на изчисление a съдържа i -тото, b съдържа $i-1$ -вото число на Фибоначи, а $n-i$ е горната граница на броя на стъпките на цикъла, които остава да бъдат изпълнени.

От условието "P е в сила преди изпълнението на оператора за цикъл", следва че трябва да построим оператор S_0 , така че да е в сила импликацията $Q \Rightarrow Wp(S_0, P)$. S_0 търсим от вида:

$a = f(n)$;

$b = g(n, a)$;

$i = h(n, a, b)$;

Имаме:

$n \geq 1 \Rightarrow f(n) = \text{fib}(h(n, f(n), g(n, f(n)))) \wedge$

$g(n, a) = \text{fib}(h(n, f(n), g(n, f(n))) - 1) \wedge$

$1 \leq h(n, f(n), g(n, f(n))) \leq n$.

Тази импликация ще е в сила, ако направим следния избор:

$f(n) = 1$, $g(n, a) = 0$ и $h(n, a, b) = 1$.

Така инициализацията получава вида:

```
S0: a=1; b=0; i=1;
```

От условието $P \wedge \neg B \Rightarrow R$ ще намерим B . Имаме:

```
a=fib(i)  $\wedge$  b=fib(i-1)  $\wedge$  1 $\leq$ i $\leq$ n  $\wedge$   $\neg B \Rightarrow$   
a=fib(n).
```

Следователно $\neg B$: $i=n$, а B : $i \neq n$. Условие 4, от списъка от условия за синтезиране на оператора за цикъл, следва от $i \leq n$ и $i \neq n$.

Тялото S на цикъла търсим от вида:

```
a=f1(n,a,b,i);  
b=g1(n,a,b,i);  
i=h1(n,a,b,i);
```

така че да са в сила условия 2 и 5 от списъка от условия за синтезиране на оператора за цикъл.

Тъй като

```
wp(t1=t; S, t<t1) =  
wp(t1=n-i; a=f1(n,a,b,i); b=g1(n,a,b,i); i=h1(n,a,b,i), n-i<t1) =  
i<h1(n,f1(n,a,b,i),g1(n, f1(n,a,b,i),b,i),i)
```

условие 5 ще е в сила, ако за $h1$ изберем $i+h$, където h е константа и $h>0$. При този избор, условие 2 има вида:

```
a=fib(i)  $\wedge$  b = fib(i-1)  $\wedge$  1 $\leq$ i $\leq$ n  $\wedge$   $i \neq n \Rightarrow$   
f1(n,a,b,i)=fib(i+h)  $\wedge$   
g1(n,f1(n,a,b,i),b,i)=fib(i+h-1)  $\wedge$  1 $\leq$ i+h $\leq$ n.
```

От $i \leq n$ и $i \neq n$, следва че за h можем да изберем 1. Тогава, според дефиницията на функцията fib , за $f1(n,a,b,i)$ и $g1(n,f1(n,a,b,i),b,i)$ избираме $a+b$ и a , съответно. Получаваме:

```
f1(n,a,b,i) = a+b;  
g1(n,a+b,b,i) = a.
```

След смяна на променливите имаме: $g1(n,a,b,i) = a-b$.

Така синтезирахме програмния фрагмент:

```
a=1; b=0; i=1;  
while (i!=n)  
{a=a+b;  
 b=a-b;  
 i++;
```

където a съдържа n -тото число на Фибоначи.

Задача 107. Дадена е числовата редица: x_0, x_1, \dots, x_{n-1} , $n \geq 2$. Да се синтезира програма, която установява дали редицата е монотонно растяща.

От условието на задачата, за предусловие и следусловие избираме:

Q: $n \geq 2$

R: $z = (\forall k: 0 \leq k \leq n-2: x_k \leq x_{k+1})$.

I начин:

Ако в R заменим n с i и укажем как i се променя, за инварианта получаваме:

P: $z = (\forall k: 0 \leq k \leq i-2: x_k \leq x_{k+1}) \wedge 2 \leq i \leq n$,

а за ограничаваща функция избираме:

t: $n-i$,

означаваща броя на останалите за сканиране елементи на редицата.

От условието "P е в сила преди изпълнението на оператора за цикъл" следва, че трябва да се намери инициализация S_0 , така че да е в сила импликацията $Q \Rightarrow Wp(S_0, P)$. S_0 търсим от вида:

$i = f_1(n)$;

$z = f_2(n, i)$.

Имаме:

$n \geq 2 \Rightarrow f_2(n, f_1(n)) = (\forall k: 0 \leq k \leq f_1(n)-2: x_k \leq x_{k+1}) \wedge 2 \leq f_1(n) \leq n$.

За да е в сила тази импликация, за $f_1(n)$ и $f_2(n, i)$ избираме 2 и $x_0 \leq x_1$, съответно. Получаваме

$S_0: i = 2; z = x[0] \leq x[1]$;

От условието $P \wedge \neg B \Rightarrow R$, т.е.

$z = (\forall k: 0 \leq k \leq i-2: x_k \leq x_{k+1}) \wedge 2 \leq i \leq n \wedge \neg B \Rightarrow$

$z = (\forall k: 0 \leq k \leq n-2: x_k \leq x_{k+1})$

намираме $\neg B: i = n \vee \neg z$ и оттук $B: (i \neq n) \wedge z$. Условие 4 има вида:

$z = (\forall k: 0 \leq k \leq i-2: x_k \leq x_{k+1}) \wedge 2 \leq i \leq n \wedge (i \neq n) \wedge z \Rightarrow$

$n-i > 0$

и е в сила. Остава да намерим тялото S на цикъла, така че да са в сила условия 2 и 5 от списъка от условия за верификация и синтез на оператора `while`. S търсим от вида:

```
i=g1(n,i,z);
z=g2(n,i,z);
```

Тъй като

$$\text{wp}(t1=t; S, t<t1) = n-g1(n,i,z) < n-i,$$

за да е в сила условие 5, за $g1(n,i,z)$ избираме $i+h$, където $h>0$.
Условие 2 има вида:

$$z=(\forall k: 0 \leq k \leq i-2: x_k \leq x_{k+1}) \wedge 2 \leq i \leq n \wedge (i \neq n) \wedge z \Rightarrow$$

$$g2(n,i+h,z)=(\forall k: 0 \leq k \leq i+h-2: x_k \leq x_{k+1}) \wedge 2 \leq i+h \leq n$$

Ако за стойности на h и $g2(n,i+h,z)$ изберем 1 и $z \wedge (x_{i-1} \leq x_i)$, съответно, това условие ще бъде в сила. Тъй като z се съдържа в лявата страна на импликацията, т.е. е `true`, $g2(n,i+1,z)$ опростяваме до $x_{i-1} \leq x_i$. Тогава $g2(n,i,z) = x_{i-2} \leq x_{i-1}$ и тялото S на оператора за цикъл има вида:

```
i++;
z=x[i-2]<=x[i-1];
```

а синтезираният програмен фрагмент:

```
int i=2;
bool z=x[0]<=x[1];
while (i!=n && z)
{
    i++;
    z=x[i-2]<=x[i-1];
}
```

където z съдържа резултата.

II начин:

Ако в R заменим 0 с i и укажем как i се променя, за инварианта получаваме:

$$P: z=(\forall k: i \leq k \leq n-2: x_k \leq x_{k+1}) \wedge 0 \leq i \leq n-2,$$

а за ограничаваща функция избираме:

$t: i.$

Търсим редица от оператори $S0$ от вида:

```
i=f1(n);
z=f2(n,i);
```

така че да е в сила импликацията $Q \Rightarrow Wp(S0, P)$. Имаме:

$$n \geq 2 \Rightarrow f2(n, f1(n)) = (\forall k: f1(n) \leq k \leq n-2: x_k \leq x_{k+1}) \wedge 0 \leq f1(n) \leq n-2$$

За да е в сила тази импликация, за $f1(n)$ избираме $n-2$, а за $f2(n, f1(n)) - x_{n-2} \leq x_{n-1}$. Получаваме

$$S0: i = n-2;$$

$$z = x[n-2] \leq x[n-1];$$

От условието $P \wedge \neg B \Rightarrow R$, т.е.

$$z = (\forall k: i \leq k \leq n-2: x_k \leq x_{k+1}) \wedge 0 \leq i \leq n-2 \wedge \neg B \Rightarrow$$

$$z = (\forall k: 0 \leq k \leq n-2: x_k \leq x_{k+1})$$

намираме $\neg B: (i=0) \vee \neg z$, а оттук $B: (i \neq 0) \wedge z$. Условие 4 има вида:

$$z = (\forall k: i \leq k \leq n-2: x_k \leq x_{k+1}) \wedge 0 \leq i \leq n-2 \wedge (i \neq 0) \wedge z \Rightarrow$$

$$i > 0$$

и очевидно е в сила. Остава да намерим тялото S на оператора за цикъл, така че да са в сила условия 2 и 5 от списъка от условия за синтез на оператора `while`. S търсим от вида:

$$i = g1(n, i, z);$$

$$z = g2(n, i, z);$$

Тъй като

$$Wp(t1=t; S, t < t1) = g1(n, i, z) < i,$$

за да е в сила условие 5, за $g1(n, i, z)$ избираме $i-h$, където $h > 0$. Условие 2 има вида:

$$z = (\forall k: i \leq k \leq n-2: x_k \leq x_{k+1}) \wedge 0 \leq i \leq n-2 \wedge (i \neq 0) \wedge z \Rightarrow$$

$$g2(n, i-h, z) = (\forall k: i-h \leq k \leq n-2: x_k \leq x_{k+1}) \wedge 0 \leq i-h \leq n-2$$

Ако за стойности на h и $g2(n, i-h, z)$ изберем 1 и $z \wedge (x_{i-1} \leq x_i)$, съответно, тази импликация ще бъде в сила. Тъй като z участва в лявата част на импликацията и следователно е в сила, $g2(n, i-1, z)$ опростяваме $x_{i-1} \leq x_i$, а оттук $g2(n, i, z) = x_i \leq x_{i+1}$. Следователно, тялото S на оператора за цикъл има вида:

$$i--;$$

$$z = x[i-1] \leq x[i];$$

а синтезираният програмен фрагмент -

```
int i=n-2;
bool z=x[n-2]<=x[n-1];
```

```

while (i!=0 && z)
{
  i--;
  z=x[i]<=x[i+1];
}

```

където z съдържа резултата.

Задача 108. Дадена е числовата редица: x_0, x_1, \dots, x_{n-1} , $n \geq 2$. Да се синтезира програма, която установява дали редицата се състои от различни елементи.

От условието на задачата, за предусловие и следусловие определяме:

Q: $n \geq 2$

R: $z = (\forall k: 0 \leq k \leq n-2: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l))$.

За инварианта избираме:

P: $z = ((\forall k: 0 \leq k \leq i-1: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l)) \wedge (\forall l: i+1 \leq l \leq j: x_i \neq x_l))$
 $\wedge 0 \leq i \leq n-2 \wedge i \leq j \leq n-1$

а за ограничаваща функция –

$$t: \frac{(n-i-1) \cdot (n-i)}{2} + n - j$$

означаваща броя на останалите двойки, за които трябва да се провери, дали са различни. Търсим инициализация S0 от вида:

```

i=f1(n);
j=f2(n,i);
z=f3(n,i,j);

```

така че импликацията $Q \Rightarrow wp(S0, P)$ да е в сила. Имаме:

$n \geq 2 \Rightarrow$

$f3(n, f1(n), f2(n, f1(n))) = ((\forall k: 0 \leq k \leq f1(n)-1: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l))$
 $\wedge (\forall l: f1(n)+1 \leq l \leq f2(n, f1(n)): x_{f1(n)} \neq x_l)) \wedge$
 $0 \leq f1(n) \leq n-2 \wedge f1(n) \leq f2(n, f1(n)) \leq n-1$.

За да е в сила тази импликация, за $f1(n)$, $f2(n,i)$ и $f3(n,i,j)$ избираме 0, 1 и $x_0 \neq x_1$, съответно. Така получаваме

S0: i=0;
 j=1;
 z=x[0] != x[1];

От условието $P \wedge \neg B \Rightarrow R$, т.е.

$$z = ((\forall k: 0 \leq k \leq i-1: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l)) \wedge (\forall l: i+1 \leq l \leq j: x_i \neq x_l)) \\ \wedge 0 \leq i \leq n-2 \wedge i \leq j \leq n-1 \wedge \neg B \Rightarrow \\ z = (\forall k: 0 \leq k \leq n-2: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l))$$

намираме $\neg B: (i=n-2 \wedge j=n-1) \vee \neg z$, а оттук $B: (i \neq n-2 \vee j \neq n-1) \wedge z$.

Условие 4 има вида:

$$z = ((\forall k: 0 \leq k \leq i-1: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l)) \wedge (\forall l: i+1 \leq l \leq j: x_i \neq x_l)) \\ \wedge 0 \leq i \leq n-2 \wedge i \leq j \leq n-1 \wedge (i \neq n-2 \vee j \neq n-1) \wedge z \Rightarrow \\ \frac{(n-i-1) \cdot (n-i)}{2} + n-j > 0$$

и е в сила (поотделно се разглеждат двата случая $i \neq n-2$ и $j \neq n-1$).

Остава да намерим тялото на цикъла S, така че да са в сила условия 2 и 5 от списъка от условия за верификация и синтез на оператора while.

S търсим от вида:

i=g1(n,i,j,z);
 j=g2(n,i,j,z);
 z=g3(n,i,j,z);

Тъй като

$$wp(t1=t; S, t < t1) =$$

$$\frac{(n-g1(n,i,j,z)-1) \cdot (n-g1(n,i,j,z))}{2} + n-g2(n,g1(n,i,j,z),j,z) < \\ \frac{(n-i-1) \cdot (n-i)}{2} + n-j$$

за да е в сила условие 5, някои възможности са:

а) $g1(n,i,j,z) = i$, $g2(n,g1(n,i,j,z),j,z) = j+1$, т.е.
 $g2(n,i,j,z) = j+1$

В този случай условие 2 има вида:

$$z = ((\forall k: 0 \leq k \leq i-1: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l)) \wedge (\forall l: i+1 \leq l \leq j: x_i \neq x_l)) \\ \wedge 0 \leq i \leq n-2 \wedge i \leq j \leq n-1 \wedge (i \neq n-2 \vee j \neq n-1) \wedge z \Rightarrow$$

$$g3(n,i,j+1,z) = ((\forall k: 0 \leq k \leq i-1: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l)) \wedge (\forall l: i+1 \leq l \leq j+1: x_i \neq x_l)) \wedge 0 \leq i \leq n-2 \wedge i \leq j+1 \leq n-1$$

То не е в сила, но ако добавим в лявата страна на импликацията предиката $j \neq n-1$ и за $g3(n,i,j+1,z)$ и изберем $z \wedge (x_i \neq x_{j+1})$, ще е в сила. Тъй като z се съдържа в лявата страна на импликацията по-горе, булевата функция $g3(n,i,j+1,z)$ можем да опростим до $x_i \neq x_{j+1}$. Тогава $g3(n,i,j,z) = x_i \neq x_j$. Следователно,

$$P \wedge B \wedge j \neq n-1 \Rightarrow wp(S1, P)$$

е в сила, където

$$S1: j++; z = x[i] \neq x[j];$$

б) $g1(n,i,j,z) = i+1$, $g2(n,g1(n,i,j,z),j,z) = g2(n,i+1,j,z) = i+2$, т.е. $g2(n,i,j,z) = i+1$

В този случай условие 2 има вида:

$$z = ((\forall k: 0 \leq k \leq i-1: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l)) \wedge (\forall l: i+1 \leq l \leq j: x_i \neq x_l))$$

$$\wedge 0 \leq i \leq n-2 \wedge i \leq j \leq n-1 \wedge (i \neq n-2 \vee j \neq n-1) \wedge z \Rightarrow$$

$$g3(n,i+1,i+2,z) = ((\forall k: 0 \leq k \leq i: (\forall l: k+1 \leq l \leq n-1: x_k \neq x_l)) \wedge$$

$$(\forall l: i+2 \leq l \leq i+2: x_{i+1} \neq x_l)) \wedge 0 \leq i+1 \leq n-2 \wedge i+1 \leq i+2 \leq n-1$$

То не е в сила, но ако добавим в лявата страна на импликацията условието $j = n-1$, тя ще е в сила, ако за $g3(n,i+1,i+2,z)$ изберем $z \wedge (x_{i+1} \neq x_{i+2})$ /лявата страна на импликацията след опростяване в този случай получава вида $P \wedge i \neq n-2 \wedge j = n-1 \wedge z$, което доказва и неравенството $i+2 \leq n-1$ /. Тъй като z се съдържа в лявата страна на импликацията, $g3(n,i+1,i+2,z)$ опростяваме до $x_{i+1} \neq x_{i+2}$, а оттук $g3(n,i,j,z) = x_i \neq x_j$. Следователно,

$$P \wedge B \wedge j = n-1 \Rightarrow wp(S2, P)$$

е в сила, където

$$S2: i++; j = i+1; z = x[i] \neq x[j].$$

От метода на преобразуващите предикати за синтезиране на оператора if/else, следва верността на импликацията:

$$P \wedge B \Rightarrow wp(S, P),$$

където

$$S: \text{if } (j \neq n-1)$$

```

{j++;
 z=x[i]!=x[j];
}
else
{i++;
 j=i+1;
 z=x[i]!=x[j];
}

```

Синтезираният програмен фрагмент има вида:

```

i=0; j=1; z=x[0]!=x[1];
while ((i!=n-2||j!=n-1) && z)
if (j!=n-1)
{j++;
 z=x[i]!=x[j];
}
else
{i++;
 j=i+1;
 z=x[i]!=x[j];
}

```

където z съдържа резултата.

Задача 109. Дадена е сортираната във възходящ ред редица от числа b_0, b_1, \dots, b_{n-1} , $n > 1$. Да се синтезира програма, която по дадено число x , така че $b_0 \leq x < b_{n-1}$, намира къде в масива b числото x може да се вмъкне, така че да се запази наредбата.

За предусловие и следусловие избираме:

Q: $n > 1 \wedge \text{sort}(b, 0..n-1) \wedge b_0 \leq x < b_{n-1}$

R: $0 \leq i < n-1 \wedge b_i \leq x < b_{i+1}$,

където параметърът i съдържа резултата, а със $\text{sort}(b, 0..n-1)$ е означен предикатът $b_0 \leq b_1 \leq \dots \leq b_{n-1}$.

За инварианта и ограничаваща функция определяме:

P: $0 \leq i < j \leq n-1 \wedge b_i \leq x < b_j \wedge \text{sort}(b, 0..n-1)$ и

t: $j-i$.

Инвариантата е получена, като в следусловието R, $i+1$ е заменено с j и е указано как j се променя, а също е добавен конюнктивен член от предусловието.

От условието "Q е в сила преди изпълнението на оператора за цикъл" следва, че трябва да се намери оператор S0, така че $Q \Rightarrow wp(S0, P)$ да е в сила. S0 търсим от вида:

$i=f(n);$

$j=g(n,i);$

Имаме:

$Q \Rightarrow wp(i = f(n); j = g(n,i), P),$ т.е.

$n>1 \wedge \text{sort}(b, 0..n-1) \wedge b_0 \leq x < b_{n-1} \Rightarrow$

$0 \leq f(n) < g(n, f(n)) \leq n-1 \wedge b_{f(n)} \leq x < b_{g(n, f(n))} \wedge \text{sort}(b, 0..n-1)$

Ако за $f(n)$ и $g(n, f(n))$ изберем 0 и $n-1$ съответно, тази импликация ще е в сила и за S0 получаваме

$i=0;$

$j=n-1;$

От условието:

$P \wedge \neg B \Rightarrow R,$ т.е.

$0 \leq i < j \leq n-1 \wedge b_i \leq x < b_j \wedge \text{sort}(b, 0..n-1) \wedge \neg B \Rightarrow$

$0 \leq i < n-1 \wedge b_i \leq x < b_{i+1},$

за B избираме $j \neq i+1$. Тъй като импликацията $P \wedge B \Rightarrow t > 0$ е в сила, ограничаващата функция t е добре избрана. Остава да намерим тялото на цикъла S, така че да са в сила условия 2 и 5 от списъка от условия за синтезиране на оператора за цикъл. S търсим от вида:

$i=f1(n,i,j);$

$j=g1(n,i,j);$

От условието:

$P \wedge B \Rightarrow wp(t1= t; S, t < t1),$ т.е.

$0 \leq i < j \leq n-1 \wedge b_i \leq x < b_j \wedge \text{sort}(b, 0..n-1) \wedge j \neq i+1 \Rightarrow$

$g1(n, f1(n,i,j), j) - f1(n,i,j) < j - i$

определяме някои възможности за $f1$ и $g1$:

$f1(n,i,j) = i+1$

$f1(n,i,j) = i$

$g1(n, f1(n,i,j), j) = j$

$g1(n, f1(n,i,j), j) = j-1$

$$f1(i, n, j) = (i+j)/2$$

$$g1(n, f1(n, i, j), j) = j$$

$$f1(n, i, j) = i$$

$$g1(n, f1(n, i, j), j) = (i+j)/2,$$

За тяло на цикъла получаваме:

S1: i++; S2: j--;;
 S3: i=(i+j)/2 S4: j=(i+j)/2;

За оператора S1, условие 2 има вида:

$$0 \leq i < j \leq n-1 \wedge b_i \leq x < b_j \wedge \text{sort}(b, 0..n-1) \wedge j \neq i+1 \Rightarrow$$

$$0 \leq i+1 < j \leq n-1 \wedge b_{i+1} \leq x < b_j \wedge \text{sort}(b, 0..n-1)$$

Тази импликация не е в сила, но единственото, което ѝ пречи е предикатът $b_{i+1} \leq x$. Ако го добавим като конюнктивен член в лявата страна на импликацията, е в сила

$$P \wedge B \wedge b_{i+1} \leq x \Rightarrow \text{wp}(S1, P).$$

За оператора S2, условие 2 има вида:

$$0 \leq i < j \leq n-1 \wedge b_i \leq x < b_j \wedge \text{sort}(b, 0..n-1) \wedge j \neq i+1 \Rightarrow$$

$$0 \leq i < j-1 \leq n-1 \wedge b_i \leq x < b_{j-1} \wedge \text{sort}(b, 0..n-1)$$

Тази импликация също не е в сила, но ако влючим в лявата страна на импликацията конюнктивния член $x < b_{i+1}$ тя ще стане в сила, т.е. ще е в сила:

$$P \wedge B \wedge x < b_{i+1} \Rightarrow \text{wp}(S2, P).$$

За целта трябва да докажем, че е от лявата страна следва $x < b_{j-1}$. Тъй като $i < j-1$ е в сила, то $i+1 \leq j-1$ и съответно $b_{i+1} \leq b_{j-1}$ са в сила. Тъй като $x < b_{i+1}$ е в сила, импликацията е доказана. От S1 и S2 образуваме условния оператор

S: if(b[i+1]<=x) i++;
 else j--;

Той ще може да е тяло на цикъла, тъй като са в сила условията от Теорема 1.

Така синтезирахме следния програмен фрагмент:

```
i=0;
j=n-1;
while(i+1!=j)
if (b[i+1]≤x) i++;
else j--;
```

който реализира линейно търсене на елемент в масив.

За оператора S3, условие 2, от списъка от условия за синтезиране на оператора за цикъл, има вида:

$$0 \leq i < j \leq n-1 \wedge b_i \leq x < b_j \wedge \text{sort}(b, 0..n-1) \wedge j \neq i+1 \Rightarrow \\ 0 \leq (i+j)/2 < j \leq n-1 \wedge b_{(i+j)/2} \leq x < b_j \wedge \text{sort}(b, 0..n-1).$$

То ще е в сила, ако в лявата страна на импликацията като конюнктивен член се добави предикатът $b_{(i+j)/2} \leq x$, т.е. в сила е:

$$P \wedge B \wedge b_{(i+j)/2} \leq x \Rightarrow \text{wp}(S3, P).$$

За S4, условие 2, от списъка от условия за синтезиране на оператора за цикъл, има вида:

$$0 \leq i < j \leq n-1 \wedge b_i \leq x < b_j \wedge \text{sort}(b, 0..n-1) \wedge j \neq i+1 \Rightarrow \\ 0 \leq i < (i+j)/2 \leq n-1 \wedge b_i \leq x < b_{(i+j)/2} \wedge \text{sort}(b, 0..n-1).$$

То ще е в сила, ако в лявата страна на импликацията като конюнктивен член се добави предикатът $x < b_{(i+j)/2}$, т.е. в сила е:

$$P \wedge B \wedge x < b_{(i+j)/2} \Rightarrow \text{wp}(S4, P).$$

Тъй като са в сила предпоставките от Теорема 1, следва, че е в сила импликацията $P \wedge B \Rightarrow \text{wp}(S, P)$, където операторът S има вида:

```
if (b[(i+j)/2] <= x) i = (i+j)/2;
else j = (i+j)/2;
```

Така синтезирахме и програмата:

```
i=0; j=n-1;
while (i+1!=j)
if (b[(i+j)/2] <= x) i = (i+j)/2;
else j = (i+j)/2;
```

която може да се опрости до:

```
i=0; j=n-1;
while (i+1!=j)
{int k=(i+j)/2;
if (b[k] <= x) i=k;
else j=k;
}
```

Тя реализира двоично търсене на елемент в редица.

Задача 110. Дадени са целочислен двумерен масив $b[0:m-1][0:n-1]$, $m>0$, $n>0$ и стойност x , която се съдържа в масива b . Да се синтезира програма, която определя мястото на първото срещане на x във b .

От условието на задачата за предусловие и следусловие определяме:

$$Q: m>0 \wedge n>0 \wedge x \in b[0:m-1][0:n-1]$$

$$R: 0 \leq i < m \wedge 0 \leq j < n \wedge x = b[i][j] \wedge x \notin b[0:i-1][0:n-1] \wedge x \notin b[i][0:j-1],$$

където i и j съдържат резултата.

Ако от R отстраним третия конюнктивен член, за инварианта получаваме:

$$P: 0 \leq i < m \wedge 0 \leq j < n \wedge x \in b[0:m-1][0:n-1] \wedge x \notin b[0:i-1][0:n-1] \wedge x \notin b[i][0:j-1].$$

За ограничаваща функция избираме:

$$t: (m-i) \cdot n - j,$$

означаваща броя на елементите на масива b , които остава да бъдат сканирани при търсене на x .

Търсим инициализация S_0 от вида:

$$i = f(m, n);$$

$$j = g(m, n, i);$$

така че да е в сила $Q \Rightarrow wp(S_0, P)$.

Имаме $m>0 \wedge n>0 \wedge x \in b[0:m-1][0:n-1] \Rightarrow$

$$0 \leq f(m, n) < m \wedge 0 \leq g(m, n, f(m, n)) < n \wedge x \in b[0:m-1][0:n-1] \wedge x \notin b[0:f(m, n)-1][0:n-1] \wedge x \notin b[f(m, n)][0:g(m, n, f(m, n))-1].$$

Тази импликация ще е в сила, ако и за $f(m, n)$, и за $g(m, n, i)$ изберем 0. Използвахме, че $b[0:-1][0:n-1]$ и $b[f(m, n)][0:-1]$ са празни множества. Следователно, S_0 има вида:

$$S_0: i=0; j=0;$$

От условието:

$$P \wedge \neg B \Rightarrow R, \text{ т.е.}$$

$$0 \leq i < m \wedge 0 \leq j < n \wedge x \in b[0:m-1][0:n-1] \wedge x \notin b[0:i-1][0:n-1] \wedge x \notin b[i][0:j-1] \wedge \neg B \Rightarrow$$

$0 \leq i < m \wedge 0 \leq j < n \wedge x = b[i][j] \wedge x \notin b[0:i-1][0:n-1] \wedge x \notin b[i][0:j-1]$
 намираме $V: x \neq b[i][j]$.

Условието $P \wedge V \Rightarrow t > 0$ има вида:

$$P \wedge V \Rightarrow (m-i) \cdot n - j > 0.$$

Изразът $(m-i) \cdot n - j$ показва колко елемента от масива b остава да бъдат сканирани при търсене на двойка (i, j) , така че $x = b[i][j]$. От верността на условието $P \wedge V$ следва, че всички сканирани до момента елементи на b са различни от x . Останалите елементи са $(m-i) \cdot n - j - 1$ на брой. Тъй като $x \in b[0:m-1][0:n-1]$, $(m-i) \cdot n - j - 1 > 0$ е в сила, условие 4 също е изпълнено. Формалното доказателство на горното твърдение, предложено от Тр. Трифонов, е приведено по-долу.

От верността на конюнкцията $P \wedge V$ следва:

$$i < m - 1 \vee (i = m - 1 \wedge j < n - 1).$$

$$a) i < m - 1$$

Следователно $m - i > 1$ е в сила. Умножаваме с $n > 0$ и получаваме $(m - i) \cdot n > n$. Но $n > j$ е в сила също, откъдето $t = (m - i) \cdot n - j > 0$.

$$b) i = m - 1 \wedge j < n - 1$$

В този случай $m - i = 1$, а $t = n - j$. От верността на V следва, че $t > 0$. Тялото S на цикъла търсим от вида:

$$i = f1(m, n, i, j);$$

$$j = g1(m, n, i, j);$$

така, че да са в сила условия 2 и 5 от списъка от условия за синтезиране на оператора `while`.

За да е в сила импликацията

$$P \wedge V \Rightarrow wp(t1 = t; S, t < t1), \text{ т.е.}$$

$$0 \leq i < m \wedge 0 \leq j < n \wedge x \in b[0:m-1][0:n-1] \wedge$$

$$x \notin b[0:i-1][0:n-1] \wedge x \notin b[i][0:j-1] \wedge x \neq b[i][j] \Rightarrow$$

$$(m - f1(m, n, i, j)) \cdot n - g1(m, n, f1(m, n, i, j)) < (m - i) \cdot n - j,$$

две възможности за $f1$ и $g1$ са:

$$f1(m, n, i, j) = i + 1$$

$$\text{и } f1(m, n, i, j) = i$$

$$g1(m, n, f1(m, n, i, j), j) = j$$

$$g1(m, n, f1(m, n, i, j), j) = j + 1.$$

От тях получаваме:

$$S1: i++ \text{ и}$$

S2: j++.

За S1, условие 2 има вида:

$$\begin{aligned} &0 \leq i < m \wedge 0 \leq j < n \wedge x \in b[0:m-1][0:n-1] \wedge x \notin b[0:i-1][0:n-1] \wedge \\ &x \notin b[i][0:j-1] \wedge x \neq b[i][j] \Rightarrow \\ &0 \leq i+1 < m \wedge 0 \leq j < n \wedge x \in b[0:m-1][0:n-1] \wedge \\ &x \notin b[0:i][0:n-1] \wedge x \notin b[i+1][0:j-1] \end{aligned}$$

Условието $i+1 < m$ ще е в сила ако поискаме да е в сила $i \neq m-1$. Условието $x \notin b[0:i][0:n-1]$ ще е в сила ако поискаме да е в сила $j = n-1$. Условието $x \notin b[i+1][0:j-1]$ ще е в сила, ако разширим S1 с оператора за присвояване $j = 0$, т.е. ако S1 има вида:

S1: i++; j=0;

В резултат имаме:

$$\begin{aligned} &0 \leq i < m \wedge 0 \leq j < n \wedge x \in b[0:m-1][0:n-1] \wedge x \notin b[0:i-1][0:n-1] \wedge \\ &x \notin b[i][0:j-1] \wedge x \neq b[i][j] \wedge i \neq m-1 \wedge j = n-1 \Rightarrow \\ &0 \leq i+1 < m \wedge 0 \leq 0 < n \wedge x \in b[0:m-1][0:n-1] \wedge x \notin b[0:i][0:n-1] \wedge \\ &x \notin b[i+1][0:-1] \end{aligned}$$

Условието $i \neq m-1$ е излишно тъй като ако допуснем, че $i = m-1$ е в сила, от $j = n-1$, ще следва, че $x \notin b[0:m-1, 0:n-1]$, което не е вярно.

Следователно

$$P \wedge B \wedge j = n-1 \Rightarrow \text{wp}(S1, P)$$

е в сила.

За оператора S2 условие 2 има вида:

$$\begin{aligned} &0 \leq i < m \wedge 0 \leq j < n \wedge x \in b[0:m-1][0:n-1] \wedge x \notin b[0:i-1][0:n-1] \wedge \\ &x \notin b[i][0:j-1] \wedge x \neq b[i][j] \Rightarrow \\ &0 \leq i < m \wedge 0 \leq j+1 < n \wedge x \in b[0:m-1][0:n-1] \wedge x \notin b[0:i-1][0:n-1] \wedge \\ &x \notin b[i][0:j] \end{aligned}$$

То ще е в сила ако в лявата страна на импликацията добавим $j \neq n-1$, т.е.

$$P \wedge B \wedge j \neq n-1 \Rightarrow \text{wp}(S2, P)$$

Тъй като са в сила предпоставките на Теорема 1, получаваме

$$P \wedge B \Rightarrow \text{wp}(S, P)$$

където операторът S има вида

```
if (j==n-1) {i++; j=0;}  
else j++;
```

Така синтезирахме програмия фрагмент:

```
i=0; j=0;  
while(x!=b[i][j])  
if(j==n-1) {i++; j=0;}  
else j++;
```

където (i, j) е позицията на първото срещане на елемента x в масива b .

Задачи

Задача 1. Да се синтезира програмен фрагмент, който проверява дали дадено естествено число е просто.

Задача 2. Да се синтезира програмен фрагмент, който намира максималния елемент на едномерния масив от числа x_0, x_1, \dots, x_{n-1} , $n \geq 1$.

Задача 3. Да се синтезира програмен фрагмент, който намира сумата от положителните и броя на отрицателните елементи на едномерния масив от числа x_0, x_1, \dots, x_{n-1} , $n \geq 1$.

Задача 4. Да се синтезира програмен фрагмент, който проверява дали числото x принадлежи на едномерния масив от числа a_0, a_1, \dots, a_{n-1} , $n \geq 1$.

Задача 5. Да се синтезира програмен фрагмент, който сортира едномерния масив от числа x_0, x_1, \dots, x_{n-1} , $n \geq 1$.

Задача 6. Да се синтезира програмен фрагмент, който слива сортираните във възходящ ред едномерни масиви от числа x_0, x_1, \dots, x_{n-1} и y_0, y_1, \dots, y_{m-1} , където $n \geq 1$ и $m \geq 1$.

Задача 7. Да се синтезира програмен фрагмент, който проверява дали числото x принадлежи на матрицата $A[n \times n]$, $n \geq 1$.

Задача 8. Да се синтезира програмен фрагмент, който проверява дали матрицата $A[n \times n]$, $n \geq 1$ е симетрична относно главния диагонал.

Задача 9. Да се синтезира програмен фрагмент, който проверява дали числото x се съдържа под главния диагонал на матрицата $A[n \times n]$, $n \geq 1$.

Задача 10. Да се синтезира програмен фрагмент, който намира сумата от елементите над главния диагонал и произведението на елементите под главния диагонал на матрицата $A[n \times n]$, $n \geq 1$.

Допълнителна литература

1. Deijkstra, E. W. A Discipline of Programming, Prentice-Hall, Englewood Cliffs, 1976.
2. Deijkstra, E. W. Guarded commands, nondeterminacy and the formal derivation of programs, Comm. ACM, v. 18, 1978.
3. Morgan, C.C. Programming from specifications, Prentice-Hall, 1990.
4. Грис, Д. Наука програмирования, Москва, Мир, 1984.
5. Манна, З. Математически основи на информатиката, София, Наука и изкуство, 1983.
6. Тодорова, М. Синтезиране на програми, СОФТЕХ, София, 1998.