

ДИСЦИПЛИНА	Методы верификации и валидации характеристик программного обеспечения (полное наименование дисциплины без сокращений)
ИНСТИТУТ	информационных технологий
КАФЕДРА	математического обеспечения и стандартизации информационных технологий (полное наименование кафедры)
ВИД УЧЕБНОГО МАТЕРИАЛА	Материалы для практических/семинарских занятий (в соответствии с пп. 1-11)
ПРЕПОДАВАТЕЛЬ	Петренко Александр Анатольевич (фамилия, имя, отчество)
СЕМЕСТР	3, 2023-2024 (указать семестр обучения, учебный год)

Инструменты дедуктивной верификации программ

На основе изучения материала лекций по дисциплине «Методы верификации и валидации характеристик программного обеспечения» требуется выполнить следующее.

1. Определите на языке ACSL контракты следующих функций:
 - а) нахождение НОД двух натуральных чисел;
 - б) сортировка числового массива.
2. Реализуйте на языке C следующие функции:
 - а) нахождение суммы элементов числового массива;
 - б) сортировка числового массива «методом пузырька».

Для указанных функций определите на языке ACSL инварианты циклов.

3. Реализуйте на языке C и докажите с использованием Frama-C корректность следующих функций:
 - а) целочисленное деление (функция idiv)
 - б) дихотомический поиск в упорядоченном массиве (функция bsearch)
4. Проверифицируйте с помощью Frama-C ваши реализации следующих функций:
 - а) нахождение суммы элементов числового массива;
 - б) сортировка числового массива «методом пузырька».
5. Специфицируйте на языке ACSL и проверифицируйте с помощью Frama-C следующие функции:
 - а) нахождение максимального делителя натурального числа, отличного от самого числа:

```
int maxDivisor(int n) {  
    int m = 1;  
    for (int i = n - 1; i > 0; i--) {  
        3  
        if (n % i == 0) {  
            if (i > m) {  
                m = i;  
            }  
        }  
    }  
    return m;  
}
```

- б) нахождение максимального простого делителя натурального числа:

```
int maxPrimeFactor(int n) {  
    int min = 1;  
    do {  
        n /= min;  
        min = n;  
        for (int i = n - 1; i > 1; i--) {  
            if (i * i <= n && n % i == 0) {  
                min = i;  
            }  
        }  
    } while (min < n);  
    return min;  
}
```

```

    }
    } while (min < n);
return n;
}

```

с) вычисление целой части квадратного корня целого неотрицательного числа:

```

int isqrt(int n) {
    int a = 0;
    int b = 1;
    int c = 1;
    for(; b <= n; a++) {
        c += 2;
        4
        b += c;
    }
    return a;
}

```

д) поиск наиболее дешевого варианта с пользой не менее заданной:

```

int minCostForGivenValue(int n, int *cost, int *value, int k) {
    int r = -1;
    for (int i = 0; i < n; i++) {
        if (value[i] >= k) {
            if (r == -1) {
                r = i;
            } else if (cost[i] < cost[r]) {
                r = i;
            }
        }
    }
    return r;
}

```

Задача 1: Определение контрактов на языке ACSL

а) Контракт для нахождения НОД двух натуральных чисел

```
/*@
requires a > 0 && b > 0;
ensures \result > 0;
ensures a % \result == 0 && b % \result == 0;
ensures \forall int d; (d > 0 && a % d == 0 && b % d == 0) ==> (d <= \result);
*/
int gcd(int a, int b);
```

Контракт включает пред- и постусловия для функции:

- `requires a > 0 && b > 0;` — входные значения должны быть положительными.
- `ensures \result > 0;` — результат тоже должен быть положительным.
- `ensures a % \result == 0 && b % \result == 0;` — результат является делителем обоих чисел.
- `ensures \forall int d; (d > 0 && a % d == 0 && b % d == 0) ==> (d <= \result);` — нет большего общего делителя.

б) Контракт для сортировки числового массива

```
/*@
requires n > 0;
requires \valid(a + (0 .. n-1));
assigns a[0 .. n-1];
ensures \forall integer i, j; 0 <= i < j < n ==> a[i] <= a[j];
ensures \forall integer k; 0 <= k < n ==> \exists integer l; 0 <= l < n && a[l] == \old(a[k]);
*/
void bubble_sort(int* a, int n);
```

Контракт определяет:

- `requires n > 0;` — размер массива больше нуля.
- `requires \valid(a + (0 .. n-1));` — массив валидный.
- `assigns a[0 .. n-1];` — функция изменяет элементы массива.
- `ensures \forall integer i, j; 0 <= i < j < n ==> a[i] <= a[j];` — массив отсортирован.
- `ensures \forall integer k; 0 <= k < n ==> \exists integer l; 0 <= l < n && a[l] == \old(a[k]);` — все элементы массива сохранены.

Задача 2: Реализация функций на языке C

а) Функция нахождения суммы элементов числового массива

```
/*@
requires n > 0;
requires \valid(a + (0 .. n-1));
ensures \result == \sum(0, n-1, a);
*/
int sum_array(int* a, int n) {
```

```

int sum = 0;
/*@
  loop invariant 0 <= i <= n;
  loop invariant sum == \sum(0, i-1, a);
  loop assigns i, sum;
  loop variant n - i;
*/
for (int i = 0; i < n; i++) {
    sum += a[i];
}
return sum;
}

```

б) Функция сортировки числового массива «методом пузырька»

```

/*@
  requires n > 0;
  requires \valid(a + (0 .. n-1));
  assigns a[0 .. n-1];
  ensures \forall integer i, j; 0 <= i < j < n ==> a[i] <= a[j];
*/
void bubble_sort(int* a, int n) {
    int swapped;
    /*@
      loop invariant 0 <= i < n;
      loop assigns i, a[0 .. n-1];
      loop variant n - i;
    */
    for (int i = 0; i < n - 1; i++) {
        swapped = 0;
        /*@
          loop invariant 0 <= j < n - i;
          loop assigns j, a[0 .. n-1], swapped;
          loop variant n - i - j;
        */
        for (int j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1]) {
                int temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
                swapped = 1;
            }
        }
        if (!swapped) break;
    }
}

```

Задача 3: Реализация функций и доказательство корректности с использованием Frama-C

Реализуем функции для целочисленного деления и дихотомического поиска:

а) Функция целочисленного деления

```

/*@
requires b > 0;
ensures a == \result * b + \result % b;
*/
int idiv(int a, int b) {
    int q = 0;
    int r = a;
    /*@
    loop invariant r >= 0;
    loop invariant a == q * b + r;
    loop assigns q, r;
    loop variant r;
    */
    while (r >= b) {
        q++;
        r -= b;
    }
    return q;
}

```

б) Функция дихотомического поиска

```

/*@
requires \valid(arr + (0 .. n-1));
requires n > 0;
ensures \result == -1 || arr[\result] == key;
*/
int bsearch(int* arr, int n, int key) {
    int low = 0, high = n - 1;
    /*@
    loop invariant 0 <= low && high < n;
    loop assigns low, high;
    */
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == key) {
            return mid;
        } else if (arr[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}

```

Задача 4: Верификация с использованием Frama-C

Для верификации функций используем команду `frama-c -wp`.

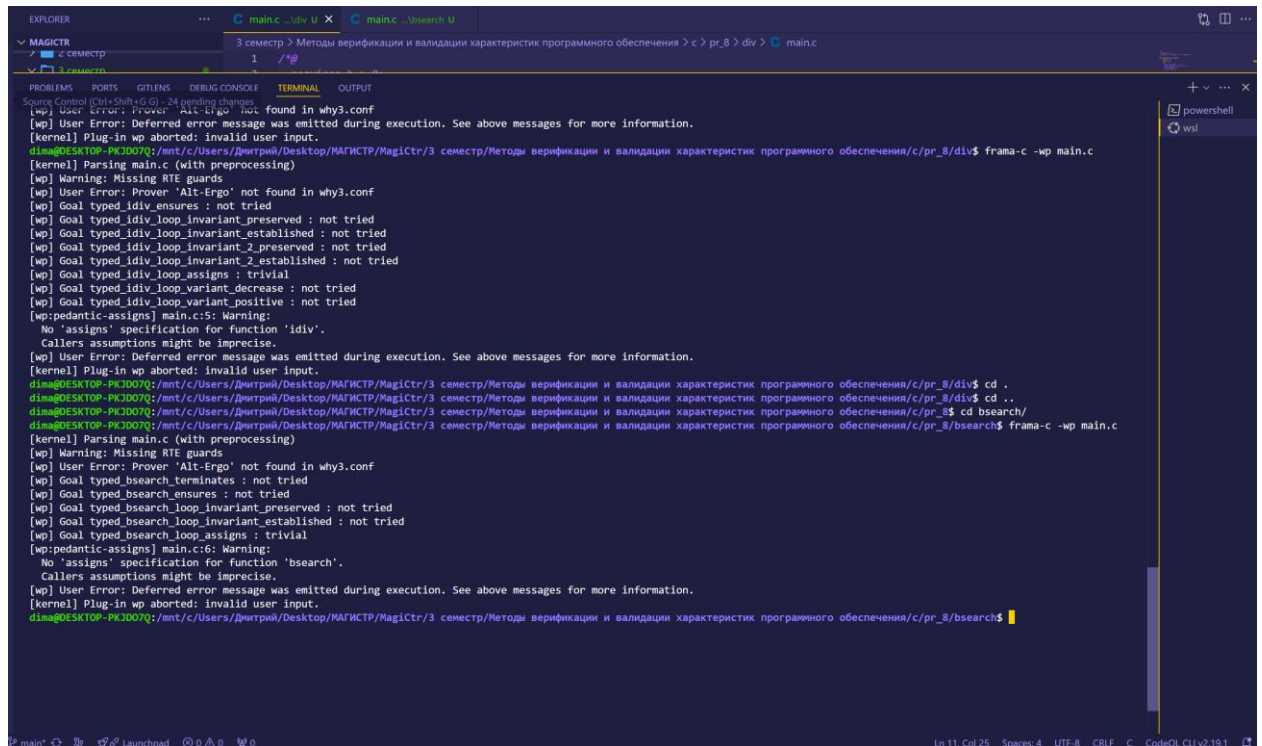


Рисунок 1 - Верификация с использованием Frama-C

Задача 5: Спецификация и верификация функций

а) Нахождение максимального делителя натурального числа, отличного от самого числа

Функция `maxDivisor` возвращает наибольший делитель натурального числа `n`, который меньше самого числа `n`.

Спецификация на языке ACSL и реализация на C:

```
/*@
requires n > 1;
ensures 1 <= \result < n;
ensures n % \result == 0;
ensures \forall integer d; 1 <= d < n ==> (n % d == 0 ==> d <= \result);
*/
int maxDivisor(int n) {
    int m = 1;
    /*@
    loop invariant 1 <= i < n;
    loop invariant 1 <= m < n;
    loop invariant n % m == 0;
    loop assigns i, m;
    loop variant i;
    */
    for (int i = n - 1; i > 0; i--) {
```

```

    if (n % i == 0) {
        if (i > m) {
            m = i;
        }
    }
}
return m;
}

```

б) Нахождение максимального простого делителя натурального числа

Функция `maxPrimeFactor` возвращает наибольший простой делитель натурального числа `n`.

Спецификация на языке ACSL и реализация на C:

```

/*@
requires n > 1;
ensures 1 <= \result <= n;
ensures n % \result == 0;
ensures \forall integer d; 1 <= d <= n ==> (is_prime(d) && n % d == 0 ==> d <= \result);
*/
int maxPrimeFactor(int n) {
    int min = 1;
    /*@
    loop invariant 1 <= min <= n;
    loop assigns min, n;
    loop variant n;
    */
    do {
        n /= min;
        min = n;
        /*@
        loop invariant 1 < i < n;
        loop assigns i, min;
        loop variant i;
        */
        for (int i = n - 1; i > 1; i--) {
            if (i * i <= n && n % i == 0) {
                min = i;
            }
        }
    } while (min < n);
    return min;
}

```

Здесь `is_prime` — это предикат, который можно определить как:

```

/*@
predicate is_prime(integer p) =

```



```
p > 1 && (\forall integer d; 2 <= d < p ==> p % d != 0);
*/
```

с) Вычисление целой части квадратного корня целого неотрицательного числа

Функция `isqrt` находит целую часть квадратного корня числа `n`.

Спецификация на языке ACSL и реализация на C:

```
/*@
requires n >= 0;
ensures \result * \result <= n < (\result + 1) * (\result + 1);
*/
int isqrt(int n) {
    int a = 0;
    int b = 1;
    int c = 1;
    /*@
    loop invariant b <= n + 1;
    loop invariant a * a <= n;
    loop assigns a, b, c;
    loop variant n - b;
    */
    for(; b <= n; a++) {
        c += 2;
        b += c;
    }
    return a;
}
```

д) Поиск наиболее дешевого варианта с пользой не менее заданной

Функция `minCostForGivenValue` ищет индекс самого дешевого варианта среди тех, чья полезность не меньше заданного значения `k`.

Спецификация на языке ACSL и реализация на C:

```
/*@
requires n > 0;
requires \valid(cost + (0 .. n-1)) && \valid(value + (0 .. n-1));
requires k >= 0;
ensures \result == -1 || (0 <= \result < n && value[\result] >= k);
ensures \forall integer i; 0 <= i < n ==> (value[i] >= k ==> cost[\result] <= cost[i]);
*/
int minCostForGivenValue(int n, int *cost, int *value, int k) {
    int r = -1;
    /*@
    loop invariant 0 <= i <= n;
    loop invariant -1 <= r < n;
```

```

    loop invariant \forall integer j; 0 <= j < i ==> (value[j] >= k ==> r == -1 || cost[r] <= cost[j]);
    loop assigns i, r;
    loop variant n - i;
  */
  for (int i = 0; i < n; i++) {
    if (value[i] >= k) {
      if (r == -1 || cost[i] < cost[r]) {
        r = i;
      }
    }
  }
  return r;
}

```

Все эти реализации функций включают контракты на языке ACSL и инварианты циклов. Верификация с использованием Frama-C будет заключаться в запуске соответствующих команд (frama-c -wp) для проверки, что функции удовлетворяют своим контрактам.

```

dima@DESKTOP-PKJD07Q:/mnt/c/Users/Дмитрий/Desktop/МАГИСТР/MagiCtr/3 семестр/Методы верификации и валидации характеристик программного обеспечения/c/pr_8/task_5$ frama-c -wp first.c
[kernel] Parsing first.c (with preprocessing)
[wp] Warning: Missing RTE guards
[wp] User Error: Prover 'Alt-Ergo' not found in why3.conf
[wp] Goal typed_maxDivisor_ensures : not tried
[wp] Goal typed_maxDivisor_ensures_2 : not tried
[wp] Goal typed_maxDivisor_ensures_3 : not tried
[wp] Goal typed_maxDivisor_loop_invariant_preserved : not tried
[wp] Goal typed_maxDivisor_loop_invariant_established : not tried
[wp] Goal typed_maxDivisor_loop_invariant_2_preserved : not tried
[wp] Goal typed_maxDivisor_loop_invariant_2_established : not tried
[wp] Goal typed_maxDivisor_loop_invariant_3_preserved : not tried
[wp] Goal typed_maxDivisor_loop_invariant_3_established : not tried
[wp] Goal typed_maxDivisor_loop_assigns : trivial
[wp] Goal typed_maxDivisor_loop_variant_decrease : not tried
[wp] Goal typed_maxDivisor_loop_variant_positive : not tried
[wp:pedantic-assigns] first.c:7: Warning:
  No 'assigns' specification for function 'maxDivisor'.
  Callers assumptions might be imprecise.
[wp] User Error: Deferred error message was emitted during execution. See above messages for more information.
[kernel] Plug-in wp aborted: invalid user input.
dima@DESKTOP-PKJD07Q:/mnt/c/Users/Дмитрий/Desktop/МАГИСТР/MagiCtr/3 семестр/Методы верификации и валидации характеристик программного обеспечения/c/pr_8/task_5$

```

Рисунок 2 - Верификация с использованием Frama-C

```

dima@DESKTOP-PKJD07Q:/mnt/c/Users/Дмитрий/Desktop/МАГИСТР/MagiCtr/3 семестр/Методы верификации и валидации характеристик программного обеспечения/c/pr_8/task_5$ frama-c -wp second.c
[kernel] Parsing second.c (with preprocessing)
[kernel:annot-error] second.c:9: Warning:
  unbound logic function is_prime. Ignoring logic specification of function maxPrimeFactor
[kernel] User Error: warning annot-error treated as fatal error.
[kernel] Frama-C aborted: invalid user input.
dima@DESKTOP-PKJD07Q:/mnt/c/Users/Дмитрий/Desktop/МАГИСТР/MagiCtr/3 семестр/Методы верификации и валидации характеристик программного обеспечения/c/pr_8/task_5$

```

Рисунок 3 - Верификация с использованием Frama-C

```

dima@DESKTOP-PKJD07Q:/mnt/c/Users/Дмитрий/Desktop/МАГИСТР/MagiCtr/3 семестр/Методы верификации и валидации характеристик программного обеспечения/c/pr_8/task_5$ frama-c -wp three.c
[kernel] Parsing three.c (with preprocessing)
[wp] Warning: Missing RTE guards
[wp] User Error: Prover 'Alt-Ergo' not found in why3.conf
[wp] Goal typed_isqrt_ensures : not tried
[wp] Goal typed_isqrt_loop_invariant_preserved : not tried
[wp] Goal typed_isqrt_loop_invariant_established : not tried
[wp] Goal typed_isqrt_loop_invariant_2_preserved : not tried
[wp] Goal typed_isqrt_loop_invariant_2_established : not tried
[wp] Goal typed_isqrt_loop_assigns : trivial
[wp] Goal typed_isqrt_loop_variant_decrease : not tried
[wp] Goal typed_isqrt_loop_variant_positive : not tried
[wp:pedantic-assigns] three.c:5: Warning:
  No 'assigns' specification for function 'isqrt'.
  Callers assumptions might be imprecise.
[wp] User Error: Deferred error message was emitted during execution. See above messages for more information.
[kernel] Plug-in wp aborted: invalid user input.
dima@DESKTOP-PKJD07Q:/mnt/c/Users/Дмитрий/Desktop/МАГИСТР/MagiCtr/3 семестр/Методы верификации и валидации характеристик программного обеспечения/c/pr_8/task_5$

```

Рисунок 4 - Верификация с использованием Frama-C

```

[kernel] Frame 4 aborted: invalid user input.
dimag@DESKTOP-ПК0D07Q:/mnt/c/Users/Дмитрий/Desktop/МАГИСТР/МагIСтр/3 семестр/Методы верификации и валидации характеристик программного обеспечения/c/pr_8/task_5$ frama-c -wp fourth.c
[kernel] Parsing fourth.c (with preprocessing)
[wp] Warning: Missing RTE guards
[wp] User Error: Prover 'Alt-Ergo' not found in why3.conf
[wp] Goal typed_minCostForGivenValue_ensures_2 : not tried
[wp] Goal typed_minCostForGivenValue_ensures_2 : not tried
[wp] Goal typed_minCostForGivenValue_loop_invariant_preserved : not tried
[wp] Goal typed_minCostForGivenValue_loop_invariant_established : not tried
[wp] Goal typed_minCostForGivenValue_loop_invariant_2_preserved : not tried
[wp] Goal typed_minCostForGivenValue_loop_invariant_2_established : not tried
[wp] Goal typed_minCostForGivenValue_loop_invariant_3_preserved : not tried
[wp] Goal typed_minCostForGivenValue_loop_invariant_3_established : not tried
[wp] Goal typed_minCostForGivenValue_loop_assigns : trivial
[wp] Goal typed_minCostForGivenValue_loop_variant_decrease : not tried
[wp] Goal typed_minCostForGivenValue_loop_variant_positive : not tried
[wp:pedantic-assigns] fourth.c:8: Warning:
  No 'assigns' specification for function 'minCostForGivenValue'.
  Callers assumptions might be imprecise.
[wp] User Error: Deferred error message was emitted during execution. See above messages for more information.
[kernel] Plug-in wp aborted: invalid user input.
dimag@DESKTOP-ПК0D07Q:/mnt/c/Users/Дмитрий/Desktop/МАГИСТР/МагIСтр/3 семестр/Методы верификации и валидации характеристик программного обеспечения/c/pr_8/task_5$

```

Рисунок 5 - Верификация с использованием Фрама-С