**Практической работы № 8.3**

**«Разработка тестовых пакетов»**

**Цель работы**

Научиться создавать тестовые пакеты с минимально достаточным количеством контрольных примеров.

**Задание**

Реализовать на любом языке программирования три программы из перечня, причём темы первых двух программ нужно выбрать по таблице 1, а третью ⁠— произвольно, исключая номера первой и второй программ. Определить классы эквивалентности. Указать и обосновать количество контрольных вариантов для тестирования.

| Распределение тем программ | |
| --- | --- |
| **Номер по журналу** | **Номера тем** |
| 1, 2 | 1, 11 |
| 3, 4 | 2, 9 |
| 5, 6 | 3, 10 |
| 7, 8 | 4, 7 |
| 9, 10 | 5, 8 |
| 11, 12 | 6, 5 |
| 13, 14 | 7, 6 |
| 15, 16 | 8, 3 |
| 17, 18 | 9, 4 |
| 19, 20 | 10, 1 |
| 21, 22 | 11, 2 |

Темы программ:

1. программа записи десятичного числа в восьмеричной системе счисления;
2. программа записи десятичного числа в шестнадцатеричной системе счисления;
3. программа записи шестнадцатеричного числа в десятичной системе счисления;
4. программа для вычисления наибольшего общего делителя двух целых чисел;
5. программа для сдвига элементов одномерного массива на указанное количество разрядов;
6. программа, реализующая сортировки элементов списка «пузырьком»;
7. программа, реализующая поиск максимального значения в двумерном массиве;
8. программа, реализующая поиск минимального значения в двумерном массиве;
9. программа, вычисляющая среднее арифметического в двумерном массиве;
10. программа, суммирующая элементы главной диагонали квадратной матрицы;
11. программа, реализующая перемешивание элементов одномерного массива.

**Обязательное техническое и программное обеспечение**

| Обязательное программное обеспечение | |
| --- | --- |
| **Тип** | **Характеристика, наименование, версия** |
| Браузер | Совместимый с HTML 5.2 и CSS Snapshot 2018 |
| Текстовый процессор | Совместимый с OpenDocument v1.0 |
| Текстовый редактор | Совместимый с UTF‑8 без маркера порядка байтов (BOM) |

Для составления исходного кода рекомендуется использовать следующие текстовые редакторы: [Visual Studio Code](https://code.visualstudio.com/), [Sublime Text](https://www.sublimetext.com/). «Блокнот», поставляемый с операционной системой *Windows*, использовать нельзя!

**Теоретические сведения**

Факторизация, или разбиение на классы эквивалентности (англ. *equivalence class partitioning (ECP)*), ⁠— техника тестирования методом чёрного ящика, которая требует разбить множество входных значений на подмножества (классы) эквивалентных элементов и разработать для каждого из подмножеств один и только один контрольный пример (англ. *test case*). Техника позволяет сократить количество контрольных примеров и, следовательно, сократить временны́е затраты на тестирование.

*ECP*, как и другие техники тестирования методом чёрного ящика, предполагают разработку контрольных примеров на основе спецификации, то есть документации на программу.

**Правила факторизации**

1. Если входное значение ограничено диапазоном m..n, то опредён один допустимый ({x|(m⩽x)∧(x⩽n)}) и два недопустимых ({a|a<m} и {b|b>n}) класса эквивалентности.
2. Если входное значение ограничено значением x, то опредён один допустимый ({x}) и один недопустимый ({y|y≠x}) класс эквивалентности.
3. Если допустимые входные значения ограничены a, m, x, то опредён один допустимый (X={a,m,x}) и один недопустимый ({y|y∉X}) класс эквивалентности.

**Пример выполнения задания**

Для отладки программ можно использовать сервис <https://www.onlinegdb.com/online_c++_compiler>.

**Пример 1. Поиск значения в массиве**

Программа реализует поиск одного значения, введённого пользователем с клавиатуры, в заранее определённом множестве или последовательности (например, в таблице базы данных). При нахождении значения выводится значение ИСТИНА, в противном случае ⁠— ЛОЖЬ.

#include <iostream>

bool find(int number, int\* sequence)

{

int length = sizeof(sequence) / sizeof(int);

for (int i = 0; i < length; i += 1) {

if (sequence[i] == number) {

return true;

}

}

return false;

}

int main()

{

using namespace std;

int element;

std::cin >> element;

int set[3] = {5, 7, 13};

cout << find(element, set) << endl;

return 0;

}

Копировать

**Решение.** Допустим, множество X={5,7,13} определено в спецификации. Тот факт, что элементы множества не повторяются по определению, а элементы последовательности могут повторяться, не имеют значения для *ECP*. Пользователь может ввести значение, как входящее во множество или последовательность X, так и не входящее в них. Следовательно, определены два класса эквивалентности: допустимый {x∈X} и недопустимый {y∉X}.

В этом случае потребуются два контрольных варианта: а) с любым значением из множества X={5,7,13}; б) ⁠с любым значением, не входящим в указанное множество.

**Пример 2. Факториал**

Дана спецификация на программу расчёта факториала. Факториал натурального числа n определяется для неотрицательных чисел как произведение всех натуральных чисел от 1 до n включительно, причём 0!=1:

n!=1×…×n=∏i=1ni.

**Задание.** Произвести разбиение входных данных на классы эквивалентности и разработать контрольные варианты.

**Решение.**

#include <iostream>

unsigned int factorial(unsigned int number) {

if (number <= 1) {

return 1;

}

return factorial(number - 1) \* number;

}

int main()

{

using namespace std;

int number;

cin >> number;

cout << factorial(number) << endl;

return 0;

}

Копировать

Разбиение зависит от спецификации.

Если спецификация ограничивает максимальное значение n натуральным числом N, т. е. N∈N, то опредён один допустимый ({x|(0⩽x)∧(x⩽N)}) и два недопустимых ({y|y<0} и {z|z>N}) класса эквивалентности. В этом случае потребуются три контрольных варианта: а) ⁠n<0; б) ⁠0⩽n⩽N; в) ⁠n>N.

Если спецификация **не** ограничивает максимальное значение n, то входное значение ограничено диапазоном 0..∞. Следовательно, опредён один допустимый ({x|(0⩽x)∧(x⩽∞)}) и только **один** недопустимый ({y|y<0}) классы эквивалентности, поскольку не существует вещественного числа r, для которого истинно r>∞.

**Примечание.** *ECP* не учитывает допустимый диапазон **выходных** данных, поэтому входное значение может приводить к недопустимому выходному значению (например, 20!=51090942171709440000 выходит за пределы типа unsigned int даже в 64­‑битных системах).

**Структура отчёта**

Отчёт о выполнении работы должен содержать следующие разделы:

* постановка задачи;
* цель работы;
* аппаратные и программные средства;
* теоретические сведения;
* входные и выходные данные (по необходимости);
* тесты (по необходимости);
* исходный код;
* результаты работы.