Мариус Бансила



Решение задач на современном C++

Станьте опытным программистом, решая практические задачи

УДК 004.4
ББК 32.973.202-018.2
Б23

Б23 Мариус Бансила

Решение задач на современном C++ / пер. с англ. А. Н. Киселева – М.: ДМК Пресс, 2019. – 302 с.: ил.

ISBN 978-5-97060-666-7

Эта книга – сборник практических задач по языку C++17: от математических и вычислительных до архитектурных, построенных на базе шаблонов проектирования. Здесь собрано 100 задач, которые помогут вам применить на практике разнообразные возможности C++ и его стандартной библиотеки, а также опробовать множество сторонних, кроссплатформенных библиотек. Решения представлены в виде исходного кода, пояснений и рекомендаций к нему.

Книга рекомендована сообществом разработчиков C++ России и Беларуси.

Издание будет полезно студентам технических вузов, а также начинающим и опытным разработчикам.

УДК 004.4
ББК 32.973.202-018.2

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-78899-386-9 (англ.)
ISBN 978-5-97060-666-7 (рус.)

Copyright © 2018 Packt Publishing.
© Оформление, перевод на русский язык, издание, ДМК Пресс, 2019

Оглавление

Вступительное слово от сообщества разработчиков России и Беларуси 12

Об авторе 13

О рецензентах 14

О чем рассказывается в книге 15

Вступление 17

Кому адресована эта книга 18

Что необходимо, чтобы извлечь максимум пользы из книги 18

Скачивание исходного кода примеров 19

Сборка примеров 19

Как сгенерировать проекты для Visual Studio 2017 20

Как сгенерировать проекты для Xcode 20

Соглашения 21

Отзывы и пожелания 22

Список опечаток 23

Нарушение авторских прав 23

Глава 1. Математические задачи 24

Задачи 24

1. Сумма натуральных чисел, кратных 3 и 5 24
2. Наибольший общий делитель 24
3. Наименьшее общее кратное 24
4. Наибольшее простое число меньше заданного 24
5. Простые числа, отличающиеся на шесть 24
6. Избыточные числа 24
7. Дружественные числа 25
8. Числа Армстронга 25
9. Простые множители числа 25
10. Код Грея 25
11. Преобразование десятичных чисел в римские 25
12. Наибольшая последовательность Коллатца 25
13. Вычисление значения числа π 25
14. Проверка действительности номеров ISBN 25

Решения 25

1. Сумма натуральных чисел, кратных 3 и 5 25
2. Наибольший общий делитель 26
3. Наименьшее общее кратное 27

4. Наибольшее простое число меньше заданного ... 27
5. Простые числа, отличающиеся на шесть ... 28
6. Избыточные числа ... 29
7. Дружественные числа ... 30
8. Числа Армстронга ... 31
9. Простые множители числа ... 32
10. Код Грея ... 33
11. Преобразование десятичных чисел в римские ... 34
12. Наибольшая последовательность Коллатца ... 36
13. Вычисление значения числа π ... 37
14. Проверка действительности номеров ISBN ... 38

Глава 2. Особенности языка ... 40

Задачи ... 40
15. Тип данных IPv4 ... 40
16. Перечисление адресов IPv4 в заданном диапазоне ... 40
17. 2-мерный массив с поддержкой базовых операций ... 40
18. Функция выбора минимального значения с переменным числом аргументов ... 40
19. Добавление диапазона значений в контейнер ... 40
20. Проверка наличия в контейнере любого, всех и ни одного из указанных значений ... 41
21. Обертка для системных дескрипторов ... 41
22. Литералы разных температурных шкал ... 41

Решения ... 41
15. Тип данных IPv4 ... 41
16. Перечисление адресов IPv4 в заданном диапазоне ... 43
17. 2-мерный массив с поддержкой базовых операций ... 45
18. Функция выбора минимального значения с переменным числом аргументов ... 47
19. Добавление диапазона значений в контейнер ... 48
20. Проверка наличия в контейнере любого, всех и ни одного из указанных значений ... 49
21. Обертка для системных дескрипторов ... 50
22. Литералы разных температурных шкал ... 54

Глава 3. Строки и регулярные выражения ... 59

Задачи ... 59
23. Преобразование чисел в строки ... 59
24. Преобразование строк в числа ... 59
25. Преобразование в верхний регистр первых букв слов ... 59
26. Объединение строк через разделитель ... 59
27. Разбиение строк на лексемы по разделителям из списка ... 60
28. Наибольшая подстрока-палиндром ... 60
29. Проверка номерного знака ... 60
30. Извлечение частей URL ... 60

31. Преобразование дат в строках....60
Решения....60
23. Преобразование чисел в строки....60
24. Преобразование строк в числа....61
25. Преобразование в верхний регистр первых букв слов....63
26. Объединение строк через разделитель....64
27. Разбиение строк на лексемы по разделителям из списка....65
28. Наибольшая подстрока-палиндром....66
29. Проверка номерного знака....68
30. Извлечение частей URL....69
31. Преобразование дат в строках....71

Глава 4. Потоки данных и файловые системы....72
Задачи....72
32. Треугольник Паскаля....72
33. Табличный вывод списка процессов....72
34. Удаление пустых строк из текстового файла....72
35. Определение размера каталога....72
36. Удаление файлов старше заданной даты....73
37. Поиск файлов в каталоге, соответствующих регулярному выражению....73
38. Временные файлы журналов....73
Решения....73
32. Треугольник Паскаля....73
33. Табличный вывод списка процессов....74
34. Удаление пустых строк из текстового файла....76
35. Определение размера каталога....77
36. Удаление файлов старше заданной даты....78
37. Поиск файлов в каталоге, соответствующих регулярному выражению....80
38. Временные файлы журналов....81

Глава 5. Дата и время....83
Задачи....83
39. Измерение времени выполнения функции....83
40. Число дней между двумя датами....83
41. День недели....83
42. День и неделя года....83
43. Время встречи для нескольких часовых поясов....83
44. Календарь на месяц....83
Решения....84
39. Измерение времени выполнения функции....84
40. Число дней между двумя датами....85
41. День недели....86
42. День и неделя года....87
43. Время встречи для нескольких часовых поясов....88
44. Календарь на месяц....90

Глава 6. Алгоритмы и структуры данных 92
Задачи 92
45. Приоритетная очередь 92
46. Циклический буфер 92
47. Двойной буфер 93
48. Самый часто встречающийся элемент в диапазоне 93
49. Текстовая гистограмма 93
50. Фильтрация списка телефонных номеров 93
51. Преобразование списка телефонных номеров 93
52. Генерация всех перестановок символов в строке 94
53. Средний рейтинг фильмов 94
54. Алгоритм объединения в пары 94
55. Алгоритм «сшивания» 94
56. Алгоритм выбора 94
57. Алгоритм сортировки 95
58. Кратчайший путь между узлами 95
59. Программа Weasel 96
60. Игра «Жизнь» 96
Решения 97
45. Приоритетная очередь 97
46. Циклический буфер 99
47. Двойной буфер 103
48. Самый часто встречающийся элемент в диапазоне 105
49. Текстовая гистограмма 107
50. Фильтрация списка телефонных номеров 108
51. Преобразование списка телефонных номеров 109
52. Генерация всех перестановок символов в строке 111
53. Средний рейтинг фильмов 113
54. Алгоритм объединения в пары 114
55. Алгоритм «сшивания» 115
56. Алгоритм выбора 117
57. Алгоритм сортировки 117
58. Кратчайший путь между узлами 121
59. Программа Weasel 125
60. Игра «Жизнь» 128

Глава 7. Конкуренция 133
Задачи 133
61. Алгоритм параллельного преобразования 133
62. Параллельные алгоритмы поиска максимального и минимального значений с использованием потоков 133
63. Параллельные алгоритмы поиска максимального и минимального значений с использованием асинхронных функций 133
64. Параллельный алгоритм сортировки 133
65. Потокобезопасное журналирование в консоль 134
66. Система обслуживания клиентов 134

Решения ... 134
61. Алгоритм параллельного преобразования ... 134
62. Параллельные алгоритмы поиска максимального и минимального значений с использованием потоков ... 136
63. Параллельные алгоритмы поиска максимального и минимального значений с использованием асинхронных функций ... 138
64. Параллельный алгоритм сортировки ... 140
65. Потокобезопасное журналирование в консоль ... 142
66. Система обслуживания клиентов ... 143

Глава 8. Шаблоны проектирования ... 148

67. Проверка пароля ... 148
68. Генерация случайных паролей ... 148
69. Генерация номеров социального страхования ... 148
70. Система одобрений ... 149
71. Контейнер с наблюдателями ... 149
72. Вычисление стоимости заказа с учетом скидок ... 149

Решения ... 150
67. Проверка пароля ... 150
68. Генерация случайных паролей ... 154
69. Генерация номеров социального страхования ... 158
70. Система одобрений ... 162
71. Контейнер с наблюдателями ... 166
72. Вычисление стоимости заказа с учетом скидок ... 171

Глава 9. Сериализация данных ... 177

Задачи ... 177
73. Сериализация и десериализация данных в формате XML ... 177
74. Выборка данных из XML с помощью XPath ... 177
75. Сериализация данных в формат JSON ... 178
76. Десериализация данных из формата JSON ... 178
77. Вывод списка фильмов в файл PDF ... 178
78. Создание документа PDF из коллекции изображений ... 179

Решения ... 179
73. Сериализация и десериализация данных в формате XML ... 179
74. Выборка данных из XML с помощью XPath ... 183
75. Сериализация данных в формат JSON ... 185
76. Десериализация данных из формата JSON ... 187
77. Вывод списка фильмов в файл PDF ... 189
78. Создание документа PDF из коллекции изображений ... 193

Глава 10. Архивы, изображения и базы данных ... 196

Задачи ... 196
79. Поиск файлов в архиве ZIP ... 196
80. Упаковка и извлечение файлов из архива ZIP ... 196
81. Упаковка и извлечение файлов из архива ZIP с защитой паролем ... 196

82. Создание файла PNG с изображением национального флага ... 196
83. Создание изображения PNG с контрольным текстом ... 197
84. Генератор штрихкодов EAN-13 ... 197
85. Чтение информации о фильмах из базы данных SQLite ... 198
86. Добавление информации о фильмах в базу данных SQLite ... 198
87. Обработка изображений для фильмов в базе данных SQLite ... 198
Решения ... 199
79. Поиск файлов в архиве ZIP ... 199
80. Упаковка и извлечение файлов из архива ZIP ... 201
81. Упаковка и извлечение файлов из архива ZIP с защитой паролем ... 204
82. Создание файла PNG с изображением национального флага ... 207
83. Создание изображения PNG с контрольным текстом ... 208
84. Генератор штрихкодов EAN-13 ... 211
85. Чтение информации о фильмах из базы данных SQLite ... 217
86. Добавление информации о фильмах в базу данных SQLite ... 223
87. Обработка изображений для фильмов в базе данных SQLite ... 227

Глава 11. Криптография ... 236

Задачи ... 236
88. Шифр Цезаря ... 236
89. Шифр Виженера ... 236
90. Кодирование и декодирование Base64 ... 236
91. Проверка учетных данных пользователя ... 236
92. Вычисление хеш-суммы файла ... 236
93. Шифрование и расшифровывание файлов ... 237
94. Подписывание файлов ... 237
Решения ... 237
88. Шифр Цезаря ... 237
89. Шифр Виженера ... 239
90. Кодирование и декодирование Base64 ... 241
91. Проверка учетных данных пользователя ... 246
92. Вычисление хеш-суммы файла ... 250
93. Шифрование и расшифровывание файлов ... 252
94. Подписывание файлов ... 254

Глава 12. Сети и службы ... 258

Задачи ... 258
95. Поиск IP-адреса хоста ... 258
96. Клиент-серверная игра Fizz-Buzz ... 258
97. Обменный курс биткойна ... 258
98. Получение почты по протоколу IMAP ... 258
99. Перевод текста на любой язык ... 259
100. Определение лиц на изображениях ... 259
Решения ... 259
95. Поиск IP-адреса хоста ... 259
96. Клиент-серверная игра Fizz-Buzz ... 261

97. Обменный курс биткойна ... 265
98. Получение почты по протоколу IMAP ... 270
99. Перевод текста на любой язык ... 274
100. Определение лиц на изображениях ... 279

Библиография ... 291
Статьи ... 291
Документация к библиотекам ... 293

Предметный указатель ... 295

Глава 1

Математические задачи

Задачи

1. Сумма натуральных чисел, кратных 3 и 5

Напишите программу, которая вычислит и выведет сумму всех натуральных чисел, кратных 3 или 5, вплоть до числа, введенного пользователем.

2. Наибольший общий делитель

Напишите программу, принимающую два положительных целых числа, которая вычислит и выведет их наибольший общий делитель.

3. Наименьшее общее кратное

Напишите программу, принимающую два положительных целых числа, которая вычислит и выведет их наименьшее общее кратное.

4. Наибольшее простое число меньше заданного

Напишите программу, которая вычислит и выведет наибольшее простое число, меньше указанного пользователем (это должно быть положительное целое число).

5. Простые числа, отличающиеся на шесть[1]

Напишите программу, которая выведет все пары простых чисел, отличающихся на шесть, вплоть до числа, введенного пользователем.

6. Избыточные числа

Напишите программу, которая выведет все избыточные числа и величину их избыточности, вплоть до числа, введенного пользователем.

---

[1] В английском языке такие пары чисел называют «sexy primes» (от латинского названия «числа шесть» – «sex»), что добавляет термину забавную двусмысленность ввиду возможной трактовки англ. «sexy primes» как «сексуальные (возбуждающие, привлекательные) простые числа». – *Прим. перев.*

7. Дружественные числа

Напишите программу, которая выведет все пары дружественных чисел меньше 1 000 000.

8. Числа Армстронга

Напишите программу, которая выведет все числа Армстронга с тремя цифрами.

9. Простые множители числа

Напишите программу, которая выведет простые множители числа, введенного пользователем.

10. Код Грея

Напишите программу, которая для всех 5-битных чисел выведет их обычное двоичное представление, представление в виде кода Грея и декодированный код Грея.

11. Преобразование десятичных чисел в римские

Напишите программу, которая для заданного десятичного значения выведет его эквивалент в виде римского числа.

12. Наибольшая последовательность Коллатца

Напишите программу, которая определит и выведет число, меньшее 1 000 000, порождающее наибольшую последовательность Коллатца, и длину этой последовательности.

13. Вычисление значения числа π

Напишите программу, вычисляющую значение числа π с точностью до двух знаков после запятой.

14. Проверка действительности номеров ISBN

Напишите программу, проверяющую действительность 10-значных номеров ISBN-10, введенных пользователем в виде строки.

Решения

1. Сумма натуральных чисел, кратных 3 и 5

Эта задача решается перебором всех чисел от 3 (1 и 2 не делятся на 3, поэтому проверять их не имеет смысла) вплоть до числа, введенного пользователем.

Подходящие числа делятся на 3 и 5 без остатка, и для этой проверки можно использовать оператор деления по модулю (%). Однако есть одна хитрость: переменная для накопления суммы должна быть объявлена с типом `long long`, а не `int` или `long`, иначе произойдет *переполнение*, если пользователь введет число больше 100 000:

```
int main()
{
  unsigned int limit = 0;
  std::cout << "Upper limit:";
  std::cin >> limit;

  unsigned long long sum = 0;
  for (unsigned int i = 3; i < limit; ++i)
  {
    if (i % 3 == 0 || i % 5 == 0)
      sum += i;
  }

  std::cout << "sum=" << sum << std::endl;
}
```

2. Наибольший общий делитель

Наибольший общий делитель (НОД) двух и более ненулевых целых чисел – это наибольшее положительное целое число, на которое все эти числа делятся без остатка. Есть несколько способов вычисления НОД, но наиболее эффективным считается алгоритм Эвклида:

```
НОД(a,0) = a
НОД(a,b) = НОД (b, a mod b)
```

Он легко реализуется на C++ в виде рекурсивной функции:

```
unsigned int gcd(unsigned int const a, unsigned int const b)
{
  return b == 0 ? a : gcd(b, a % b);
}
```

А вот так выглядит нерекурсивная реализация алгоритма Эвклида:

```
unsigned int gcd(unsigned int a, unsigned int b)
{
  while (b != 0) {
    unsigned int r = a % b;
    a = b;
    b = r;
  }
```

```
  return a;
}
```



В C++17 имеется функция `constexpr gcd()`, объявленная в заголовке `<numeric>`, которая вычисляет наибольший общий делитель двух чисел.

3. Наименьшее общее кратное

Наименьшее общее кратное (НОК) двух и более ненулевых целых чисел – это наименьшее положительное целое число, кратное всем этим числам. Одно из решений задачи поиска наименьшего общего кратного сводится к задаче поиска наибольшего общего делителя, согласно следующей формуле:

```
НОК(a, b) = abs(a, b) / НОД(a, b)
```

Вот как выглядит функция вычисления наименьшего общего кратного:

```
int lcm(int const a, int const b)
{
  int h = gcd(a, b);
  return h ? (a * (b / h)) : 0;
}
```

Для вычисления НОК для трех и более чисел можно использовать алгоритм `std::accumulate` из заголовка `<numeric>`:

```
template<class InputIt>
int lcmr(InputIt first, InputIt last)
{
  return std::accumulate(first, last, 1, lcm);
}
```



В C++17 имеется функция `constexpr lcm()`, объявленная в заголовке `<numeric>`, которая вычисляет наименьшее общее кратное двух чисел.

4. Наибольшее простое число меньше заданного

Простыми называют числа, которые делятся без остатка только на самих себя и на 1. Чтобы найти наибольшее простое число меньше заданного, вы должны написать функцию, определяющую – является ли число простым. А затем

вызывать ее для каждого числа, начиная с заданного и заканчивая 1, пока не встретится первое простое число. Существуют разные алгоритмы определения простого числа. Вот один из них:

```
bool is_prime(int const num)
{
  if (num <= 3) { return num > 1; }
  else if (num % 2 == 0 || num % 3 == 0)
  {
    return false;
  }
  else
  {
    for (int i = 5; i * i <= num; i += 6)
    {
      if (num % i == 0 || num % (i + 2) == 0)
      {
        return false;
      }
    }
    return true;
  }
}
```

А вот как можно использовать эту функцию для решения задачи:

```
int main()
{
  int limit = 0;
  std::cout << "Upper limit:";
  std::cin >> limit;

  for (int i = limit; i > 1; i--)
  {
    if (is_prime(i))
    {
      std::cout << "Largest prime:" << i << std::endl;
      return 0;
    }
  }
}
```

5. Простые числа, отличающиеся на шесть

Простые числа, отличающиеся на шесть, – это пары целых чисел, разность которых равна шести (например, 5 и 11 или 13 и 19). Существуют также *простые близнецы*, отличающиеся на два, *простые двоюродные числа*, отличающиеся на четыре.

В решении предыдущей задачи мы реализовали функцию, которая выясняет, является ли целое число простым. Ее можно использовать в решении данной задачи. Для этого нужно просто проверить, являются ли простыми числа *n* и *n*+6, и вывести их:

```
int main()
{
  int limit = 0;
  std::cout << "Upper limit:";
  std::cin >> limit;

  for (int n = 2; n <= limit; n++)
  {
    if (is_prime(n) && is_prime(n+6))
    {
      std::cout << n << "," << n+6 << std::endl;
    }
  }
}
```

В качестве дополнительного упражнения можете попробовать найти и вывести тройки простых чисел, отличающихся на шесть, четверки и пятерки.

6. Избыточные числа

Избыточным называют число, сумма положительных собственных делителей которого превышает само число. Собственные делители числа – это положительные простые делители, отличные от самого числа. Величина, на которую сумма собственных делителей превышает само число, называется его избыточностью. Например, число 12 имеет собственные делители 1, 2, 3, 4 и 6. Их сумма равна 16, то есть число 12 является избыточным. Его избыточность равна 4 (то есть 16 – 12).

Чтобы найти сумму всех собственных делителей, нужно опробовать все числа от 2 до квадратного корня числа (все простые делители меньше или равны этому значению). Если текущее число, назовем его *i*, делит число *num* без остатка, тогда оба числа, *i* и *num*/*i*, являются делителями. Но если они равны (например, если *i* = 3 и *num* = 9, тогда *i* делит 9, но *n*/i = 3), в сумму добавляется только *i*, потому что каждый делитель должен быть добавлен только один раз. Иначе в сумму добавляются оба числа, *i* и *num*/*i*:

```
int sum_proper_divisors(int const number)
{
  int result = 1;
  for (int i = 2; i <= std::sqrt(number); i++)
  {
    if (number%i == 0)
    {
```

```
        result += (i == (number / i)) ? i : (i + number / i);
      }
    }
    return result;
  }
```

Вывод избыточных чисел реализуется как простой цикл до указанного предела, вычисляющий сумму собственных делителей и сравнивающий ее с самим числом:

```
void print_abundant(int const limit)
{
  for (int number = 10; number <= limit; ++number)
  {
    auto sum = sum_proper_divisors(number);
    if (sum > number)
    {
      std::cout << number << ", abundance="
      << sum - number << std::endl;
    }
  }
}

int main()
{
  int limit = 0;
  std::cout << "Upper limit:";
  std::cin >> limit;

  print_abundant(limit);
}
```

7. Дружественные числа

Два числа называют дружественными, если сумма собственных делителей одного из них равна другому числу. Собственные делители числа – это положительные простые делители, отличные от самого числа. Не путайте *дружественные числа* (amicable numbers) с *компанейскими числами* (friendly numbers). Например, число 220 имеет собственные делители 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 и 110, сумма которых равна 284. Число 284 имеет собственные делители 1, 2, 4, 71 и 142; их сумма равна 220. То есть числа 220 и 284 являются дружественными.

Задача решается перебором всех чисел до заданного предела. Для каждого числа определяется сумма его собственных делителей. Назовем ее `sum1`. Затем процесс повторяется, и определяется сумма собственных делителей числа `sum1`. Если результат получился равным первому числу, значит, это число и `sum1` – дружественные числа:

```
void print_amicables(int const limit)
{
  for (int number = 4; number < limit; ++number)
  {
    auto sum1 = sum_proper_divisors(number);
    if (sum1 < limit)
    {
      auto sum2 = sum_proper_divisors(sum1);
      if (sum2 == number && number != sum1)
      {
        std::cout << number << "," << sum1 << std::endl;
      }
    }
  }
}
```

Функция `sum_proper_divisors()`, используемая в этом решении, взята из решения задачи об избыточных числах.



Функция выше выведет пары чисел дважды, например «220,284» и «284,220». Измените реализацию, чтобы каждая пара выводилась только один раз.

8. Числа Армстронга

Числа Армстронга (названные в честь Майкла Ф. Армстронга (Michael F. Armstrong)), их также называют *самовлюбленными числами* и *совершенными цифровыми инвариантами*, – это числа, равные сумме своих цифр, возведенных в степень, равную количеству цифр. Например, наименьшее число Армстронга – 153, которое равно $1^3 + 5^3 + 3^3$.

Чтобы определить, является ли трехзначное число самовлюбленным, нужно сначала разбить его на цифры, чтобы потом подсчитать сумму их степеней. Однако для этого требуется использовать довольно дорогостоящие операции деления. Намного проще положиться на тот факт, что число – это сумма цифр, умноженных на 10 в степени, соответствующей позиции цифры. Иными словами, для чисел до 1000 мы имеем: `a*10^2 + b*10^1 + c`. Поскольку по условиям задачи требуется найти только числа Армстронга с тремя цифрами, можно утверждать, что `a` начинается с 1. Это решение будет работать быстрее других, потому что умножения выполняются компьютерами быстрее, чем деление. Вот как выглядит реализация нужной нам функции:

```
void print_narcissistics()
{
```

```
  for (int a = 1; a <= 9; a++)
  {
    for (int b = 0; b <= 9; b++)
    {
      for (int c = 0; c <= 9; c++)
      {
        auto abc = a * 100 + b * 10 + c;
        auto arm = a * a * a + b * b * b + c * c * c;
        if (abc == arm)
        {
          std::cout << arm << std::endl;
        }
      }
    }
  }
}
```

В качестве дополнительного упражнения можете написать функцию, которая будет определять, является ли число самовлюбленным, независимо от количества цифр в нем. Такая функция может получиться довольно медленной, потому что сначала ей придется определить последовательность цифр числа, сохранить их в контейнере, а затем сложить их степени (по числу цифр).

9. Простые множители числа

Простые множители положительного целого числа – это простые числа, которые делят данное число без остатка. Например, простые множители числа 8 – это 2 × 2 × 2, а простые множители числа 42 – это 2 × 3 × 7. Для определения простых множителей можно использовать следующий алгоритм:

1. Если n делится на 2, 2 – простой множитель и его следует добавить в список, после этого, пока возможно, n продолжает делиться на 2. По завершении этого шага у нас остается нечетное число n.
2. Выполнить перебор чисел от 3 до квадратного корня из n. Если текущее число, назовем его i, делит n без остатка и является простым, его следует добавить в список, и у нас остается результат деления n/i. После этого, пока возможно, n продолжает делиться на i, и затем i увеличивается на 2 (чтобы получить следующее нечетное число).
3. Если n – простое число, большее 2, шаги выше не приведут к превращению n в 1. Следовательно, если в конце шага 2 число n остается больше 2, значит, оно является простым множителем.

```
std::vector<unsigned long long> prime_factors(unsigned long long n)
{
  std::vector<unsigned long long> factors;
  while (n % 2 == 0) {
```

```
    factors.push_back(2);
    n = n / 2;
  }
  for (unsigned long long i = 3; i <= std::sqrt(n); i += 2)
  {
    while (n%i == 0) {
      factors.push_back(i);
      n = n / i;
    }
  }

  if (n > 2)
    factors.push_back(n);
  return factors;
}

int main()
{
  unsigned long long number = 0;
  std::cout << "number:";
  std::cin >> number;

  auto factors = prime_factors(number);
  std::copy(std::begin(factors), std::end(factors),
    std::ostream_iterator<unsigned long long>(std::cout, " "));
}
```

В качестве дополнительного упражнения определите наибольший простой множитель числа 600 851 475 143.

10. Код Грея

Код Грея, также известный как *рефлексивный двоичный код*, – это форма двоичного кодирования, в которой две соседние кодовые комбинации отличаются только одним битом. Кодирование в рефлексивный двоичный код выполняется по следующей формуле:

```
if b[i-1] = 1 then g[i] = not b[i]
else g[i] = b[i]
```

Она эквивалентна такому выражению:

```
g = b xor (b логический сдвиг вправо на 1)
```

Для декодирования рефлексивного двоичного кода используется следующая формула:

```
b[0] = g[0]
b[i] = g[i] xor b[i-1]
```

На языке C++ это можно записать (с использованием 32-битных беззнаковых целых чисел) так:

```
unsigned int gray_encode(unsigned int const num)
{
  return num ^ (num >> 1);
}

unsigned int gray_decode(unsigned int gray)
{
  for (unsigned int bit = 1U << 31; bit > 1; bit >>= 1)
  {
    if (gray & bit) gray ^= bit >> 1;
  }
  return gray;
}
```

Вывести все 5-битные целые числа с их двоичными представлениями, кодами Грея и декодированными значениями можно с помощью следующего кода:

```
std::string to_binary(unsigned int value, int const digits)
{
  return std::bitset<32>(value).to_string().substr(32-digits, digits);
}

int main()
{
  std::cout << "Number\tBinary\tGray\tDecoded\n";
  std::cout << "------\t------\t----\t-------\n";

  for (unsigned int n = 0; n < 32; ++n)
  {
    auto encg = gray_encode(n);
    auto decg = gray_decode(encg);

    std::cout
      << n << "\t" << to_binary(n, 5) << "\t"
      << to_binary(encg, 5) << "\t" << decg << "\n";
  }
}
```

11. Преобразование десятичных чисел в римские

Римские числа, известные в наши дни, записываются с помощью семи символов: I = 1, V = 5, X = 10, L = 50, C = 100, D = 500 и M = 1000. Система использует операции сложения и вычитания при составлении числовых символов. Символы от 1 до 10: I, II, III, IV, V, VI, VII, VIII, IX и X. У римлян не было символа, пред-