Д. Д. Бычкова
Московский государственный областной университет, Россия

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОБУЧЕНИЮ ПРОЦЕССУ СОСТАВЛЕНИЯ АЛГОРИТМОВ С ИХ ДАЛЬНЕЙШЕЙ РЕАЛИЗАЦИЕЙ НА ЯЗЫКАХ ПРОГРАММИРОВАНИЯ

Аннотация
Сегодня наше общество достигло такого уровня развития, что во многих сферах человеческой деятельности разные виды работ выполняют роботы. Особенно это касается тех сфер, которые связаны с опасностью для жизни человека или требуют серьезных физических затрат. Но роботы — это пока всего лишь исполнители, а алгоритмы и программы, по которым функционируют роботы, пишет человек. Он занимает основную позицию в тандеме человек—робот. Для составления алгоритмов разного уровня сложности необходима совокупность теоретических знаний и практических умений и навыков, которые формируются как в процессе освоения теоретических основ, так и в процессе непрерывной практической деятельности по применению теории к реализации различного рода задач. В статье описываются процессы составления алгоритмов для двух практических задач разного уровня сложности, даются пояснения и рекомендации, а также приводятся фрагменты программ, реализованных по данным алгоритмам на языке программирования С++.
Ключевые слова: алгоритм, одномерный массив, структура, строковый тип данных, язык программирования С++.
DOI: 10.32517/2221-1993-2020-19-3-31-36

Ежедневно каждый человек решает разного рода задачи в различных сферах его деятельности, в том числе в повседневной жизни. Задача может быть простой, а может быть сложной. Простую задачу решать, естественно, легче, над сложной необходимо серьезно подумать (иногда не один год), но и в том, и в другом случае для достижения результата человек должен выполнить определенный набор действий. Это может быть давно известный набор действий, а может быть набор действий, сформированный человеком здесь и сейчас, исходя из поставленной задачи. Если каждое действие из данного набора будет понятно и определено, то человек, выполнив все действия, достигнет результата быстрее, если нет, то ему придется разбираться с каждым действием отдельно, для чего потребуются определенные знания, умения и время.

Набор четко определенных и понятных действий, позволяющий достичь поставленной цели, есть ***алгоритм****.*

Сегодня много алгоритмов выполняют за человека роботы, компьютеры, автоматы, но они являются всего лишь исполнителями, а сами алгоритмы, по которым действуют машины, пишет человек.

Для составления сложных алгоритмов необходимы определенные знания, умения и навыки, которые формируются не столько в процессе освоения теоретических основ, сколько в процессе обязательной и непрерывной практической деятельности. Чем больше практики и погружения в процесс составления алгоритмов, тем более ясным и понятным становится этот процесс.

Изучение алгоритмов можно начать со знакомства с уже существующими алгоритмами, например, с алго-

Контактная информация

Бычкова Дарья Дмитриевна, канд. пед. наук, доцент кафедры вычислительной математики и методики преподавания информатики физико-математического факультета, Московский государственный областной университет, Россия; *адрес:* 141014, Россия, Московская обл., г. Мытищи, ул. Веры Волошиной, д. 24; *e-mail:* dd.bychkova@mgou.ru

D. D. Bychkova,
Moscow Region State University, Russia

GUIDELINES FOR TRAINING THE PROCESS OF CREATING ALGORITHMS WITH THEIR FURTHER IMPLEMENTATION ON PROGRAMMING LANGUAGES

Abstract
Today our society has reached such a level of development that in many areas of human activity, it is robots that perform different types of work. This is especially true for those areas that are associated with a danger to human life or require serious physical costs. But robots are still only performers. Algorithms, programs by which robots function, is written by a person. Thus, man occupies a major position in the human—robot tandem. To compile algorithms of different levels of complexity, a combination of theoretical knowledge and practical skills is needed, which are formed both in the process of mastering the theoretical foundations and in the process of continuous practical activity on applying the theory to the implementation of various kinds of tasks.The article presents guidelines for creating infographics for students under the guidance of a teacher in extracurricular computer science classes using special computer programs. The article presents guidelines for creating infographics for students under the guidance of a teacher in extracurricular computer science classes using special computer programs.
Keywords: algorithm, one-dimensional array, structure, string data type, C ++ programming language.

ритмами сортировки, поиска и др. [1]. Проанализировать их, понять, как они работают, выявить особенности, оценить сложность. Следующим этапом должно быть применение этих алгоритмов на практике для реализации конкретной задачи. Затем можно попробовать доработать один из существующих алгоритмов. После этого нужно пробовать составить свой алгоритм и реализовать его в программном коде [3].

Все вышеперечисленное является цепочкой действий, выполнение каждого из которых должно медленно, но верно приближать к определенному результату, а именно к формированию алгоритмического мышления в довольно серьезных областях знаний, и, по сути, это тоже является алгоритмом.

Начинать формировать знания, умения и навыки в области составления алгоритмов и учить понимать данный процесс необходимо уже в школе. Учащиеся обязательно должны составлять алгоритм решения задачи (словесное описание по пунктам, блок-схему), а не просто реализовывать решение на языке программирования. Далее необходимо анализировать алгоритм, уточнять, рационализировать, а затем выделять те пункты (блоки), которые могут пригодиться в дальнейшем при решении других задач.

Рассмотрим процесс составления алгоритмов для двух задач разной сложности. Данные примеры демонстрируют, **как поэтапно можно разрабатывать алгоритмы для решения задач на первых этапах обучения составлению алгоритмов.**

Задача 1.

Пусть есть следующая задача [2]:

Дан массив целых чисел. Найти сумму элементов, кратных заданному числу.

После того как задача поставлена, необходимо осуществить **анализ ее условия**, т. е. рассмотреть каждую его составляющую и определить список вопросов, ответы на которые позволят сформировать представление о том, как задачу возможно реализовать.

Анализ условия позволил сформулировать следующие **вопросы:**

1. Что такое одномерный массив?
2. Как объявить массив и указать количество его элементов?
3. Как заполнить одномерный массив?
4. Имеет ли смысл выводить одномерный массив на экран?
5. Откуда появляется число, которому кратны элементы массива?
6. Какие элементы являются кратными указанному числу?
7. Как найти сумму элементов массива?

Ответим на вышеуказанные вопросы, сохраняя при этом их нумерацию:

1. Одномерный массив — это пронумерованная последовательность элементов одинакового типа.
2. Объявление массива предполагает, что компилятору сообщаются: имя массива, количество элементов, из которых он состоит, и тип этих элементов.

   Пример объявления массива: int a[10] — это одномерный массив: имя массива — a, массив состоит из 10 элементов, каждый из которых есть целое число (int).

   Для указания компилятору количества элементов, из которых состоит массив, можно использовать квалификатор const:

```
const int n =10;
int a[n];
```

3. Заполнить одномерный массив можно несколькими способами:
   - ввести элементы с клавиатуры;
   - сгенерировать элементы случайным образом;
   - инициализировать элементы массива при его объявлении.
4. Да, нужно выводить одномерный массив на экран — это дает возможность оценить, верно ли считается сумма. Но целесообразно выводить массив на экран только в том случае, если его элементы инициализируются при объявлении или генерируются случайным образом, так как при заполнении элементов одномерного массива с клавиатуры они и так отображаются на экране и дополнительный вывод будет лишним.
5. Число, которому кратны элементы массива, можно вводить с клавиатуры (обозначим его P и будем в дальнейшем использовать это обозначение).
6. Элементы одномерного массива, кратные числу P, это те элементы, которые делятся нацело на P, т. е. при делении элемента одномерного массива на P остаток от деления равен 0.
7. Сумма элементов одномерного массива находится последовательным прибавлением элемента к предыдущей сумме (изначально сумма равна нулю).

Учитывая развернутые ответы, полученные на вопросы, **можно сформировать первоначальный алгоритм, который затем при необходимости можно будет уточнить**:

1. Объявить одномерный массив, задав количество его элементов с помощью квалификатора const (пусть $n = 10$).
2. Организовать цикл для заполнения одномерного массива случайным образом (например, из диапазона от –4 до 10).
3. Организовать цикл для вывода элементов одномерного массива на экран.
4. Ввести число P.
5. Организовать цикл для нахождения суммы элементов одномерного массива, кратных P.
6. Вывести сумму элементов массива на экран.

Теперь **проанализируем пункты данного алгоритма.**

Очевидно, что пункты 2 и 3 можно объединить внутри одного цикла, а пункт 5 расписать более подробно, указав, каким образом будет осуществляться поиск кратных элементов одномерного массива.

Таким образом, получим следующий **уточненный алгоритм:**

1. Объявить одномерный массив, задав количество его элементов с помощью квалификатора const (пусть $n = 10$).
2. Организовать цикл для:
   2.1. заполнения одномерного массива случайным образом (например, из диапазона от –4 до 10);
   2.2. вывода элементов одномерного массива на экран.
3. Ввести число *P*.
4. Организовать цикл, внутри которого будет осуществляться проверка условия и выполнение соответствующего действия: если элемент одномерного массива делится на введенное с клавиатуры число *P* и остаток от деления в этом случае равен 0, то осуществить суммирование таких элементов.
5. Вывести сумму элементов массива на экран.

Осуществим еще раз анализ пунктов последнего полученного алгоритма.

В пунктах 2 и 4 осуществляется работа с элементами одномерного массива: ввод элемента, вывод элемента на экран, проверка элемента на кратность. Не имеет смысла использовать в программе два одинаковых цикла — можно осуществлять проверку на кратность сразу после того, как элемент был сгенерирован случайным образом. Тогда ввод числа *P* должен осуществляться *перед* заполнением массива.

Добавим также еще один пункт — инициализацию нулем переменной *s*, в которой хранится сумма элементов массива.

После всех добавлений и изменений получаем следующий алгоритм:

1. Объявить одномерный массив, задав количество его элементов с помощью квалификатора const (пусть $n = 10$).
2. Ввести число *P*.
3. Инициализировать переменную *s* нулем ($s = 0$).
4. Организовать цикл для:
   4.1. заполнения одномерного массива случайным образом (например, из диапазона от –4 до 10);
   4.2. проверки условия и выполнения соответствующего действия:
   если элемент одномерного массива делится на введенное с клавиатуры число *P* и остаток от деления в этом случае равен 0, то осуществить суммирование таких элементов;
   4.3. вывода элементов одномерного массива на экран.
5. Вывести сумму элементов массива на экран.

Данный алгоритм является универсальным, и его можно запрограммировать практически на любом языке программирования.

На рисунке 1 представлен **фрагмент программы на языке программирования C++**, написанной по вышеуказанному алгоритму, в комментариях дано подробное описание каждой строки и указаны соответствующие данным строкам пункты алгоритма.

Задача 2.

Теперь рассмотрим более сложную задачу [4]:

На вход программе подаются сведения о годовых осадках (миллиметры атмосферных осадков) в городах Московской области не более чем за десятилетний период. В первой строке сообщается о количестве записей (не более 99), во второй строке вводится начальный год периода, в третьей строке — конечный год периода, каждая из следующих N строк имеет формат: <Город> <год> <осадки>, где <Город> — строка, состоящая не более чем из 20 символов, <год> — не более чем четырехзначный номер, <осадки> — целое число. <Город> <год> <осадки> разделены одним пробелом. Пример первых входных строк:

10
2005
2008

```
…
const int n = 10;                //объявление количества элементов массива (10 элементов) с помощью
                                   квалификатора const(пункт 1)
  int arr[n];                    //объявление массива (пункт 1)
  int P;                         //объявление переменной P
  cout << "Введите число P, которому должны быть кратны элементы одномерного массива ";  //вывод на
                                   экран сообщения для пользователя
  cin >> P;                      //ввод числа P (пункт 2)
  int s = 0;                     //объявление и инициализация переменной s (пункт 3)
  for (int i = 0; i < n; i++)    //организация цикла (пункт 4) для:
  {
    arr[i] = rand() % 15 - 4;    //генерирования случайным образом элементов одномерного массива
                                   (пункт 4.1)
    if (arr[i] % P == 0)         //проверки условия (пункт 4.2): если элемент массива делится
                                   на введенное с клавиатуры число P и остаток от деления равен 0, то:
       s += arr[i];              //суммирование таких элементов (пункт 4.2)
    cout << setw(4) << arr[i]; //вывода элементов массива на экран (пункт 4.3)
  }
  cout << "\n Сумма элементов одномерного массива, кратных P, равна " << s;  //вывод на экран
                                   сообщения для пользователя и результата (суммы) (пункт 5)
…
```

Рис. 1. Фрагмент листинга программы для решения задачи 1

Подольск 2006 650
Мытищи 2006 700
Наро-Фоминск 2005 650

Требуется написать программу, которая выводит на экран информацию о городах с минимальным и максимальным уровнем осадков в каждом году (если городов с максимальными/минимальными значениями в году несколько, то вывести все эти города).

Проанализировав задачу, можно составить следующий **список вопросов, ответы на которые помогут при составлении алгоритма:**

1. Как получить *N* записей?
2. Как получить начальный год периода?
3. Как получить конечный год периода?
4. Как задать строку: <Город> <год> <осадки>?
5. Как найти максимальный уровень осадков?
6. Как найти минимальный уровень осадков?
7. Как найти максимальный/минимальный уровень осадков в определенном году?

Ответим на вышеуказанные вопросы, сохраняя их номера:

1. Количество записей, каждая из которых представляет собой строку с информацией <Город> <год> <осадки>, должно быть введено с клавиатуры (не более 99 записей — ограничение, указанное в условии задачи).
2. Начальный год периода вводится с клавиатуры и представляет собой четырехзначное число.
3. Конечный год периода вводится с клавиатуры и представляет собой четырехзначное число.
4. Строка: <Город> <год> <осадки> вводится с клавиатуры столько раз, сколько предполагается записей (см. п. 1).

   В условии задачи указано, что <Город>, <год> и <осадки> разделены одним пробелом, но нигде не указано конкретно, что это должна быть единая и неделимая изначально строка. Поэтому ввод данной строки можно осуществлять двумя способами: 1) ввести единую строку или 2) ввести все три части этой строки отдельно. В первом случае строку придется делить на три отдельные составляющие: <Город>, <год> и <осадки>, чтобы была возможность работать с каждой составляющей отдельно, во втором случае, наоборот, необходимо будет их объединить для того, чтобы в конце выводилась уже единая строка.

   Данный пункт является наиболее сложным моментом при составлении алгоритма. Если вводится единая строка, то ее обязательно надо разделить, чтобы извлечь необходимые данные и работать уже с ними. Но процесс разделения именно переменной строкового типа на составляющие — довольно сложный процесс в любом языке программирования. Менее сложным является обратный процесс — объединение, но в этом случае необходимо добавлять третий одномерный массив, каждый элемент которого является элементом строкового типа. И тот и другой варианты могут быть реализованы, однако в разных языках программирования есть свои особенности для работы с записями, которые могут быть использованы для решения рассматриваемой задачи. Так как в данном случае предполагается программирование решения задачи на языке C++, то целесообразно применить структуру, которая изначально представляет собой набор переменных, объединенных общим именем, и с каждой переменной которой, называемой членом, можно работать независимо от других [5].
5. Найти максимальный уровень осадков проще всего следующим образом: занести все осадки в одномерный массив и осуществить в нем поиск максимального элемента.
6. Минимальный уровень осадков целесообразно искать аналогично п. 5.
7. Найти максимальный/минимальный уровень осадков в *определенном году* можно таким образом:
   - создать два одномерных массива;
   - занести годы в первый одномерный массив, при этом каждый номер элемента-года должен соответствовать номеру соответствующей строки, вводимой изначально;
   - занести осадки во второй одномерный массив, каждый номер элемента которого будет соответствовать номеру элемента первого одномерного массива, в котором находятся элементы-годы;
   - осуществить поиск максимального/минимального элемента во втором одномерном массиве с учетом соответствующего условия для первого одномерного массива (больше или равно начальному году периода, но меньше или равно конечному году периода).

Так как далее предполагается, что программа будет реализована на языке программирования C++ с использованием структуры, то дополнительно создавать одномерные массивы не требуется — каждый член структуры уже будет являться самостоятельным одномерным массивом.

Учитывая все вышесказанное, составим сначала **алгоритм, который может быть принят за основу при программировании на любом языке программирования**. При этом выберем вариант раздельного ввода данных и их объединения.

Алгоритм будет выглядеть следующим образом:

1. Ввести количество записей.
2. Ввести начальный год периода.
3. Ввести конечный год периода.
4. Организовать цикл для:
   4.1. ввода названия города (строковый тип);
   4.2. ввода года (целое число);
   4.3. ввода числа осадков (целое число);
   4.4. объединения названия города, года и числа осадков в единую строку;
   4.5. внесения даты (года) в первый одномерный массив;
   4.6. внесения числа осадков во второй одномерный массив;
   4.7. проверки условия и выполнения соответствующих действий:

если введенный год больше или равен начальному году периода и меньше или равен конечному году периода, то:

4.7.1. осуществить поиск максимального числа осадков среди всех элементов второго одномерного массива;

4.7.2. осуществить поиск минимального числа осадков среди всех элементов второго одномерного массива.

5. Организовать цикл для:

5.1. проверки условия и выполнения соответствующих действий:
если введенный год больше или равен начальному году периода и меньше или равен конечному году периода, то:

5.1.1. осуществить проверку условия и выполнение соответствующего действия:
если число осадков равно максимальному или минимальному значению, то вывести запись с соответствующими данными на экран.

Теперь **конкретизируем алгоритм для его реализации на языке программирования C++:**

1. Объявить структуру (члены структуры: переменная-город, переменная-год, переменная-осадки).
2. Ввести количество записей.
3. Ввести начальный год периода.
4. Ввести конечный год периода.

```
…
struct Towns //объявление структуры (пункт 1)
{
  string town;
  int year;
  int rain;
};
…
  Towns a[100];                                  //объявление переменной структурного типа
  …
  cout << "Введите количество записей: ";       //сообщение, выводимое на экран
  cin >> N;                                      //ввод количества записей (пункт 2)
  cout << "Введите начальный год: ";            //сообщение, выводимое на экран
  cin >> year_begin;                             //ввод начального года периода (пункт 3)
  cout << "Введите конечный год: ";             //сообщение, выводимое на экран
  cin >> year_end;                               //ввод конечного года периода (пункт 4)
  …
  for (int i = 0; i < N; i++)                    //организован цикл (пункт 5) для:
  {
    cin >> a[i].town;                            //ввода названия города (пункт 5.1)
    cin >> a[i].year;                            //ввода года (пункт 5.2)
    cin >> a[i].rain;                            //ввода числа осадков (пункт 5.3)
    if (a[i].year >= year_begin && a[i].year <= year_end)   //проверки условия (пункт 5.4): если
                                                 введенный год больше или равен начальному году
                                                 периода и меньше или равен конечному году периода, то:
      {
        if (a[i].rain > max) max = a[i].rain;              //поиск максимального значения числа
                                                             осадков (пункт 5.4.1)
        if (a[i].rain < min) min = a[i].rain;              //поиск минимального значения числа
                                                             осадков (пункт 5.4.2)
      }
  }
  for (int i = 0; i < N; i++)                              //организован цикл (пункт 6) для:
  {
    if (a[i].year >= year_begin && a[i].year <= year_end)  //проверки условия (пункт 6.1): если
                                                 введенный год больше или равен начальному году
                                                 периода и меньше или равен конечному году периода, то:
      if ((a[i].rain == min) || (a[i].rain == max))
       cout << a[i].town<<" " << a[i].year<<" "<<a[i].rain << endl;  //проверка условия и выполне-
                                        ние соответствующего действия (пункт 6.1.1): если число
                                        осадков равно максимальному или минимальному значению,
                                        то вывод записи с соответствующими данными на экран
  }
  …
```

Рис. 2. Фрагмент листинга программы для решения задачи 2

5. Организовать цикл для:
   5.1. ввода названия города (строковый тип);
   5.2. ввода года (целое число);
   5.3. ввода числа осадков (целое число);
   5.4. проверки условия и выполнения соответствующих действий:
   если введенный год больше или равен начальному году периода и меньше или равен конечному году периода, то:
      5.4.1. осуществить поиск максимального числа осадков среди всех;
      5.4.2. осуществить поиск минимального числа осадков среди всех.
6. Организовать цикл для:
   6.1. проверки условия и выполнения соответствующих действий:
   если введенный год больше или равен начальному году периода и меньше или равен конечному году периода, то:
      6.1.1. осуществить проверку условия и выполнение соответствующего действия:
      если число осадков равно максимальному или минимальному значению, то вывести запись с соответствующими данными на экран.

На рисунке 2 представлен **фрагмент программы на языке программирования C++**, соответствующий вышеуказанному алгоритму, с подробными комментариями в каждой строке.

* * *

Реализация рассмотренных выше двух практических задач разного уровня сложности позволяет продемонстрировать процессы поэтапного составления алгоритмов, которые способствуют формированию алгоритмической культуры обучающихся.

Список использованных источников

1. *Бхаргава А.* Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. СПб.: Питер, 2020. 288 с.

2. *Златопольский Д. М.* Сборник задач по программированию. СПб.: БХВ-Петербург, 2011. 304 с.

3. Как лучше всего изучать алгоритмы — отвечают эксперты. https://tproger.ru/experts/how-to-learn-algorithms/

4. *Самылкина Н. Н., Островская Е. М., Кузнецова Е. Ю.* ЕГЭ 2013. Информатика: тренировочные задания. М.: Эксмо, 2012. 200 с.

5. *Шилдт Г.* C++: полное руководство, классическое издание. СПб.: ООО «Диалектика», 2019. 800 с.

ПРАВИЛА ДЛЯ АВТОРОВ

Уважаемые коллеги!

Статьи для публикации в журналах «Информатика и образование» и «Информатика в школе» должны отправляться в редакцию **только через электронную форму на сайте ИНФО (раздел «Авторам → Отправка статьи»)**:

http://infojournal.ru/authors/send-article/

Обращаем ваше внимание, что для отправки статьи необходимо предварительно зарегистрироваться на сайте ИНФО (или авторизоваться — для зарегистрированных пользователей).

С требованиями к оформлению представляемых для публикации материалов можно ознакомиться на сайте ИНФО в разделе «Авторам»:

http://infojournal.ru/authors/

Обратите внимание: требования к оформлению файла рукописи — **разные** для журналов «Информатика и образование» и «Информатика в школе». При подготовке файла рукописи ориентируйтесь на требования для того журнала, в который вы представляете статью. Если вы представляете рукопись в оба журнала (для публикации в одном из изданий — на усмотрение редакции), при ее оформлении следует руководствоваться требованиями к оформлению рукописи в журнал «Информатика и образование».

Дополнительную информацию можно получить в разделе **«Авторам → Часто задаваемые вопросы»**:

http://infojournal.ru/authors/faq/

а также в редакции ИНФО:

E-mail: readinfo@infojournal.ru

Телефон: (495) 140-19-86