Н. Ю. Добровольская, А. В. Харченко,
Кубанский государственный университет, г. Краснодар

ЗАДАНИЯ-ИССЛЕДОВАНИЯ КАК СПОСОБ РАЗВИТИЯ НАВЫКОВ ПРОГРАММИРОВАНИЯ

Аннотация

В работе раскрывается актуальность формирования научно-исследовательской компетенции в старших классах школы и на первом курсе вуза в рамках изучения информатики. Предлагается для достижения поставленной цели выполнение заданий-исследований по программированию. Приводятся структура исследования, ряд примеров, соответствующих базовым темам курса «Информатика» и инвариантных к выбору языка программирования. Выполнение заданий позволяет учащемуся сформулировать обобщенную схему решения задач некоторого типа. Использование заданий-исследований в педагогической практике способствует формированию у обучаемого целостного восприятия конструкций языка программирования и соответствующих структур данных. По мере освоения базовых тем дисциплины учащийся получает в свое распоряжение различные механизмы решения задач. Предлагаемыми заданиями-исследованиями можно расширить методику обучения информатике.

Ключевые слова: исследование, научно-исследовательская компетенция, методика обучения информатике, техника алгоритмизации, системный подход к решению задач по программированию, схема решения задачи.

DOI: 10.32517/2221-1993-2019-18-3-48-51

Одной из составляющих учебного курса информатики в старших классах и на первых курсах профильных направлений подготовки в вузах является раздел «Программирование». Изучение техники алгоритмизации, применение некоторого языка программирования для решения задач чаще всего сводятся к составлению программ заданий возрастающей сложности [1–3]. Учащимся последовательно предлагаются к освоению различные конструкции языка и структуры данных: условный оператор, операторы цикла, массивы, матрицы, файлы, списки и т. д. Однако процесс программирования — это прежде всего творческий, исследовательский процесс. Одна и та же задача может решаться различными способами, с применением разных структур данных. Раскрытие многообразия подходов к решению задачи с первых занятий через использование в процессе обучения заданий-исследований, на наш взгляд, является актуальной педагогической задачей при изучении программирования. Формировать умение подбирать различные эвристики решения задачи, навыки сравнения и анализа алгоритмов целесообразно уже при освоении основ программирования.

Задание-исследование предполагает четкую постановку задачи, но выбор структуры данных, основных используемых конструкций языка программирования, алгоритма решения должен быть обоснован [6, 9]. Учащемуся необходимо рассмотреть несколько вариантов решения задачи, указать преимущества и недостатки каждого из них, области возможного применения.

Задание-исследование имеет следующую структуру:

- цель исследования;
- этапы исследования;
- варианты решения на некотором языке программирования;
- наборы тестовых решений;
- выводы.

Применение заданий-исследований на практических занятиях по программированию решает следующие педагогические задачи:

Контактная информация

Добровольская Наталья Юрьевна, канд. пед. наук, доцент, доцент кафедры информационных технологий, Кубанский государственный университет, г. Краснодар; *адрес:* 350040, г. Краснодар, ул. Ставропольская, д. 149; *e-mail:* dnu10@mail.ru

Харченко Анна Владимировна, ст. преподаватель кафедры информационных технологий, Кубанский государственный университет, г. Краснодар; *адрес:* 350040, г. Краснодар, ул. Ставропольская, д. 149; *e-mail:* fz@mail.ru

N. Yu. Dobrovolskaya, A. V. Kharchenko,
Kuban State University, Krasnodar

RESEARCH TASKS AS A METHOD OF DEVELOPMENT OF PROGRAMMING SKILLS

Abstract

The article reveals the relevance of the formation of research competence in high school and the first year of university in the study of informatics. It is proposed to achieve the goal while performing research tasks on programming. The structure of the study is described, a number of examples invariant to the choice of programming language is given. Performing tasks allows the student to formulate a generalized scheme for solving problems of a certain type. The use of research tasks in teaching practice contributes to the formation of the student's holistic perception of programming language structures and corresponding data structures. As mastering the basic themes of the discipline, the student masters various mechanisms for solving problems. The proposed research tasks can expand the method of teaching informatics.

Keywords: research, research competence, teaching methods to informatics, algorithmic techniques, systematic approach to solving programming tasks, problem solving scheme.

- формирование у учащихся исследовательской компетенции;
- развитие навыков программирования;
- обеспечение системного подхода к решению задач по программированию;
- повышение положительной мотивации к изучению дисциплины [11, 12, 16].

Мы предлагаем ряд заданий-исследований, апробированных на практических занятиях со студентами первого курса факультета компьютерных технологий и прикладной математики Кубанского государственного университета [5, 13, 14]. Опыт использования подобных заданий переносим на уроки информатики в старших классах образовательных школ.

Задание 1.

Постановка задачи.

Дано *N* целых чисел. Необходимо найти наибольший элемент и указать его порядковый номер.

Цель исследования: определить наиболее эффективную структуру данных для решения задачи.

Этапы исследования.

1. Реализовать алгоритм решения с использованием массива и дополнительной переменной для хранения порядкового номера наибольшего элемента.
2. Реализовать алгоритм решения с использованием массива, но без дополнительной переменной для порядкового номера.
3. Реализовать алгоритм решения без использования массива.

Варианты решений на C++.

Вариант 1.

```
void main()
{int a[100];
int n;
cout<<"enter the size of array:"; cin>>n;
for (int i=0; i<n; i++)
 cin>>a[i];
int max=1000, ind=0;
for (i=0; i<n; i++)
  if (a[i]>max) {max=a[i]; ind=i;};
cout<<ind;
}
```

Вариант 2.

```
int ind=0;
for (i=0; i<n; i++)
  if (a[i]>a[ind]) ind=i;
```

Вариант 3.

```
void main()
{int x, n;
int max=1000; ind=0;
cin>>n;
for (int i=0; i<n; i++)
{ cin>>x;
  if (x>max) {max=x; ind=i;};
}
cout<<ind;
}
```

Выводы: рассмотренная задача допускает поиск ответа за одно прохождение набора данных, а следовательно, использование массива данных для хранения исходных чисел не обязательно и является неэффективным по отношению к использованию памяти.

Любое задание-исследование в зависимости от уровня обучаемого может предлагаться только в виде постановки задачи и цели исследования или более полно с раскрытием этапов исследования [7, 8, 15].

Результатом выполнения задания-исследования является **обобщенная схема:**

```
ввод n количества элементов набора чисел
max=достаточно большое число
for i от 1 до n
{
  ввод числа x из набора
  if (x>max) {max=x; ind=i;}
}
```

Эта схема может быть расширена на ситуацию, когда количество элементов в наборе заранее неизвестно. Окончанием ввода чисел служит некий маркер, например, число 0. Тогда схема решения задачи будет иметь вид:

```
ввод числа x
i=1;
max=x; ind=0;
while (x не равно 0)
{
  if (x>max) {max=x; ind=i;}
  ввод числа x из набора
  i=i+1;
}
```

Задание 2.

Постановка задачи.

Дано *N* целых чисел. Необходимо найти количество наибольших элементов набора.

Цель исследования: определить наиболее эффективную структуру данных для решения задачи.

Этапы исследования.

1. Реализовать алгоритм решения с использованием массива и двух проходов по массиву данных.
2. Реализовать алгоритм решения без использования массива.

Варианты решений на C++.

Вариант 1.

```
void main()
{int a[100];
int n;
cout<<"enter the size of array:"; cin>>n;
for (int i=0; i<n; i++)
cin>>a[i];
int max=1000; k=0;
for (i=0; i<n; i++)
  if (a[i]>max) max=a[i];
for (i=0; i<n; i++)
  if (a[i]==max) k++;
cout<<k;
}
```

Вариант 2.

```
void main()
{int x, n;
int max=1000; k=0;
cin>>n;
for (int i=0; i<n; i++)
{ cin>>x;
  if (x>max) {max=x; k=0;}
  if (x==max) {k++;}
}
cout<<k;
}
```

Выводы: вариант 2 позволяет не хранить исходные данные в массиве, а следовательно, эффективно использовать выделенную память.

Задание 3.

Постановка задачи.

Дано N целых чисел. Необходимо найти наибольший элемент набора, а затем найти второй наибольший элемент (т. е. найти максимальный элемент после вычеркивания первого найденного наибольшего значения).

Цель исследования: определить наиболее эффективную структуру данных для решения задачи.

Этапы исследования.

1. Реализовать алгоритм решения с использованием массива и двух проходов по набору данных.
2. Реализовать алгоритм решения без использования массива и с одним проходом по исходному набору данных.

Варианты решений на C++.

Вариант 1.

```
void main()
{int a[100];
int n;
cout<<"enter the size of array:"; cin>>n;
for (int i=0; i<n; i++)
cin>>a[i];
int max1=1000; max2=1000;
for (i=0; i<n; i++)
  if (a[i]>max1) max1=a[i];
for (i=0; i<n; i++)
  if (a[i]>max2 && a[i]<max1) max2=a[i];
cout<<max1<<max2;
}
```

Вариант 2.

```
void main()
{int x, n;
int max1=1000; max2=1000;
cin>>n;
for (int i=0; i<n; i++)
{ cin>>x;
  if (x>max1) {max2=max1; max1=x;}
  if (x>max2 && x<max1) max2=x;
}
cout<<max1<<max2;
}
```

Выводы: второй вариант решения является более эффективным, так как в нем не используется статическая структура данных (массив) и решение представлено одним циклом.

Задание 4.

Постановка задачи.

Дана квадратная матрица порядка n. Необходимо найти сумму четных элементов, расположенных на главной диагонали матрицы.

Цель исследования: определить количество операторов цикла, необходимых для решения задачи, позволяющих использовать меньшее число обращений к элементам.

Этапы исследования.

1. Реализовать алгоритм решения с использованием двух операторов цикла для обработки матрицы.
2. Реализовать алгоритм решения с использованием одного оператора цикла для обработки матрицы.

Варианты решений на C++.

Вариант 1.

```
void main()
{int a[10][10];
int n;
cout<<"enter the size of array:"; cin>>n;
for (int i=0; i<n; i++)
 for (int j=0; j<n; j++)
cin>>a[i][j];
int s=0;
for (i=0; i<n; i++)
 for (int j=0; j<n; j++)
  if (a[i][j]%2 == 0 && i==j) s+=a[i][j];
cout<<s;
}
```

Вариант 2.

```
int a[10][10];
int n;
cout<<"enter the size of array:"; cin>>n;
for (int i=0; i<n; i++)
 for (int j=0; j<n; j++)
cin>>a[i][j];
int s=0;
for (i=0; i<n; i++)
  if (a[i][i]%2 == 0) s+=a[i][i];
cout<<s;
}
```

Выводы: вариант 2 позволяет сократить число обращений к элементам матрицы, а следовательно, сокращает время работы программы.

Задание 5.

Постановка задачи.

Дана квадратная матрица порядка n. Необходимо найти количество положительных элементов, расположенных выше главной диагонали матрицы.

Цель исследования: определить вид операторов цикла, необходимых для решения задачи, позволяющих использовать меньшее число обращений к элементам.

Этапы исследования.

1. Реализовать алгоритм решения с полным просмотром всех элементов матрицы.
2. Реализовать алгоритм решения с просмотром только элементов, лежащих в требуемой области (выше главной диагонали).

Варианты решений на C++.

Вариант 1.

```
void main()
{int a[10][10];
int n;
cout<<"enter the size of array:"; cin>>n;
for (int i=0; i<n; i++)
 for (int j=0; j<n; j++)
cin>>a[i][j];
int k=0;
for (i=0; i<n; i++)
 for (int j=0; j<n; j++)
  if (a[i][j] > 0 && i < j) k++;
cout<<k;
}
```

Вариант 2.

```
int a[10][10];
int n;
cout<<"\nenter the size of array:"; cin>>n;
for (int i=0; i<n; i++)
 for (int j=0; j<n; j++)
cin>>a[i][j];
int k=0;
for (i=0; i<n; i++)
 for (int j=i+1; j<n; j++)
  if (a[i][j] > 0 && i < j) k++;
cout<<k;
}
```

Выводы: в варианте 2 включение условия области расположения элементов матрицы в структуру вложенных циклов сокращает время работы программы.

* * *

Первые три задания можно предлагать обучаемым сразу после изучения темы «Массивы». Эти задания являются основой многих алгоритмов задания 27 ЕГЭ по информатике и ИКТ. Четвертое и пятое задания предлагается рассматривать после освоения темы «Матрицы».

Задания-исследования можно использовать в курсе информатики при обучении программирования как в старших классах школ, так и на первом курсе вуза для студентов, обучающихся по ИТ-направлениям [4, 10]. Реализация заданий не привязана к конкретному языку программирования и допускает написание решений на языках Pascal, Basic, C++, Python. Педагог может предлагать задания-исследования на занятиях, в качестве заданий самостоятельной работы, на факультативных занятиях, выполнять исследования можно индивидуально или малой группой.

Набор заданий легко расширяем. В общем случае достаточно взять задачу, допускающую либо несколько алгоритмов решения (задача сортировки, поиск элемента с заданными свойствами и т. д.), либо использование нескольких структур данных (например, поиск заданного элемента среди набора чисел реализуем посредством массива, файла, списка). Предполагаемое направление исследования (алгоритмы, используемые структуры данных) может разбираться с обучаемым перед выполнением задания. Или учащийся должен самостоятельно предложить этапы исследования. Аргументацию наиболее эффективного решения обучаемый приводит самостоятельно — именно это позволяет сформировать исследовательскую компетенцию.

Применение заданий-исследований на практических занятиях по программированию способствует формированию у обучающегося целостного восприятия конструкций языка программирования и соответствующих структур данных. По мере освоения базовых тем дисциплины учащийся получает в свое распоряжение различные механизмы решения задач. В нашем случае одна из основных педагогических целей — сформировать у обучаемого навык самостоятельного подбора эффективных эвристик решения той или иной задачи.

Список использованных источников

1. *Грушевский С. П., Добровольская Н. Ю.* Курс «Информационные технологии в науке и образовании» в процессе формирования профессионально-педагогических компетенций магистрантов математических направлений // Труды Международной научной конференции «Образование, наука и экономика в вузах и школах. Интеграция в международное образовательное пространство». Цахкадзор: Pedagogic initiative, 2014. Т. 1. С. 489–492.

2. *Грушевский С. П., Добровольская Н. Ю.* Проектирование профессионально-педагогической подготовки студентов математических направлений на основе технологий формирования их IT-компетенций // Известия Алтайского государственного университета. Серия: Педагогика и психология. 2013. № 2/1 (78). С. 18–22.

3. *Добровольская Н. Ю.* Формирование умения формального исполнения алгоритма как основы алгоритмических навыков учащихся // Преподавание математики и информатики в школе и вузе. Материалы межвузовской научно-практической конференции. Краснодар, 2017. С. 56–58.

4. *Добровольская Н. Ю., Харченко А. В.* Применение информационных технологий в обучении // Актуальные проблемы информационно-правового пространства. Сборник статей по материалам ежегодных Всероссийских научно-практических конференций. Краснодар, 2017. С. 28–31.

5. *Добровольская Н. Ю., Харченко А. В.* Применение технологии фасетов при изучении основ программирования // Математическое образование в школе и вузе: теория и практика (MATHEDU-2014): Материалы IV Международной научно-практической конференции, посвященной 210-летию Казанского университета и Дню математики (г. Казань, 28–29 ноября 2014 года). Казань: Изд-во Казан. ун-та, 2014. С. 231–233.

6. *Егошина И. Л.* Методология научных исследований: учебное пособие. Йошкар-Ола: ПГТУ, 2018. 148 с.

7. *Казаринова И. Н.* Методологический практикум. Сборник упражнений по основам методологии и методики научных исследований: учебно-практическое пособие: в 4 ч. М.; Берлин: Директ-Медиа, 2018. Ч. 1. 77 с.

8. *Костюк Н. В.* Методы исследования в профессиональном образовании: учебно-методическое пособие. Кемерово: Кемеровский государственный институт культуры, 2016. 92 с.

9. *Леонова О. В.* Основы научных исследований: методические рекомендации для практических занятий. М.: Альтаир-МГАВТ, 2015. 62 с.

10. Методология педагогического исследования: практикум / сост. Н. В. Колосова. Ставрополь: СКФУ, 2017. 102 с.

11. *Мусина О. Н.* Основы научных исследований: учебное пособие. М.; Берлин: Директ-Медиа, 2015. 150 с.

12. *Новиков В. К.* Методология и методы научного исследования: курс лекций. М.: Альтаир-МГАВТ, 2015. 211 с.

13. *Харченко А. В.* Методика обучения будущих учителей конструированию учебных задач по информатике на основе фасетной технологии // Известия ВГПУ. Серия: Педагогические науки. 2016. № 2 (271). С. 89–92.

14. *Харченко А. В., Добровольская Н. Ю.* Фасетная технология как способ построения наборов учебных задач // Известия ВГПУ. Серия: Педагогические науки. 2016. № 1 (270). С. 53–57.

15. *Шкляр М. Ф.* Основы научных исследований: учебное пособие. М.: Дашков и К°, 2017. 208 с.

16. *Юдина О. И.* Методология педагогического исследования: учебное пособие. Оренбург: ОГУ, 2013. 141 с.