Автор: Некрасов С.А., Тазетдинова Ю.А., Тазетдинов Б.И.
13.11.2025 11:40 - Обновлено 28.11.2025 11:06

АЛГОРИТМЫ ОБРАБОТКИ МАССИВОВ КАК ОСНОВА ФОРМИРОВАНИЯ АЛГОРИТМИЧЕСКОГО МЫШЛЕНИЯ: МЕТОДИКА ПОСТРОЕНИЯ ПРАКТИКУМА ДЛЯ НАЧИНАЮЩИХ

С.А. Некрасов, студент 1 курса ПИ

Ю.А. Тазетдинова, к.ф.-м.н., доцент

Б.И. Тазетдинов, к.ф.-м.н., доцент

г. Бирск, Бирский филиал Уфимского университета науки и технологий

Аннотация. В работе рассматривается методика построения практикума по изучению алгоритмов обработки массивов для начинающих программистов. Предлагается системный подход к формированию алгоритмического мышления через последовательное освоение базовых операций с массивами. Методика основана на принципе «от простого к сложному» и включает четыре ключевых этапа: от работы с отдельными элементами до реализации сложных алгоритмов сортировки и поиска. Особое внимание уделяется развитию навыков декомпозиции задач, анализу временной сложности и формированию устойчивых паттернов мышления, необходимых для решения практических задач программирования.

Ключевые слова: алгоритмическое мышление, массивы, обработка массивов, алгоритмы поиска, алгоритмы сортировки, сложность алгоритмов, декомпозиция задач.

Введение. Современное программирование предъявляет высокие требования к качеству и эффективности программного обеспечения, где алгоритмическое мышление становится фундаментальным навыком успешного разработчика. Однако у начинающих программистов переход от освоения синтаксиса языка к решению практических алгоритмических задач часто сопровождается значительными трудностями. Студенты, уверенно владеющие базовыми конструкциями языка C#, нередко оказываются не готовы к декомпозиции сложных задач, анализу эффективности решений и выбору оптимальных алгоритмов для конкретных проблем.

Особую актуальность эта проблема приобретает при изучении массивов – одной из базовых структур данных, которая служит идеальной платформой для формирования алгоритмического мышления. Традиционный подход к обучению, сосредоточенный преимущественно на синтаксисе работы с массивами, оказывается недостаточно эффективным для развития у студентов способности самостоятельно проектировать и оптимизировать алгоритмы.

В данной статье предлагается методика построения практикума, который последовательно развивает алгоритмическое мышление через систему специально разработанных заданий по обработке массивов. Предлагаемый подход основан на принципе постепенного усложнения – от элементарных операций с отдельными

элементами массива до реализации сложных алгоритмов поиска и сортировки. Каждый этап практикума не только знакомит студентов с конкретными алгоритмами, но и формирует универсальные навыки анализа сложности, сравнения альтернативных решений и оптимизации кода.

Особое внимание уделяется формированию устойчивых паттернов мышления, которые позволяют студентам переносить полученные навыки на решение более сложных задач программирования. Методика обеспечивает плавный переход от понимания конкретных алгоритмов работы с массивами к освоению общих принципов алгоритмизации, что создает прочную основу для дальнейшего профессионального роста в области разработки программного обеспечения.

1. Базовые операции с элементами массива. Практикум начинается с фундаментальных операций, которые формируют понимание структуры массива и принципов работы с индексами. Например, задача нахождения суммы элементов массива:

```
int[] numbers = { 2, 5, 8, 12, 15 };

int sum = 0;

for (int i = 0; i < numbers.Length; i++)

{
```

```
    sum  += numbers[i];

}

Console.WriteLine($"Сумма элементов:  {sum}");
```

Перед написанием кода студенты должны проговорить алгоритм словами, затем записать его в виде псевдокода, и только после этого переходить к реализации на C#.

2. Поисковые алгоритмы. На этом этапе вводится понятие сложности алгоритмов и сравниваются различные подходы к решению поисковых задач. Линейный поиск:

```
int FindIndex(int[]  array, int target)

{

    for (int i = 0; i < array.Length; i++)

    {

        if  (array[i] == target)
```

```
            return  i;

    }

    return -1;

}
```

Рекомендуется анализ лучшего, худшего и среднего случаев. Обсуждение необходимости оптимизации и условий, при которых линейный поиск является оптимальным решением.

3. Алгоритмы агрегации. Студенты учатся выявлять закономерности в данных и преобразовывать массивы. Поиск максимального элемента и его индекса:

```
int FindMaxIndex(int[] array)

{

    int  maxIndex = 0;
```

```
    for (int i = 1; i < array.Length; i++)

    {

        if  (array[i] > array[maxIndex])

            maxIndex  = i;

    }

    return maxIndex;

}
```

Рекомендуется решение вариативных задач (поиск второго по величине элемента, подсчет элементов, удовлетворяющих условию) для развития гибкости мышления.

4. Алгоритмы сортировки. Изучение простейших алгоритмов сортировки закладывает основу для понимания более сложных структур данных:

```
void BubbleSort(int[]  array)
```

```
{

    for (int i = 0; i < array.Length - 1; i++)

    {

        for  (int j = 0; j < array.Length - i - 1;  j++)

        {

            if  (array[j] > array[j + 1])

            {

                int  temp = array[j];

                array[j] = array[j + 1];

                array[j + 1] = temp;

            }

        }
```

```
    }


}
```

Рекомендуется визуализация процесса сортировки, анализ количества сравнений и перестановок, обсуждение эффективности алгоритма в различных ситуациях.

Заключение. Предложенная методика построения практикума позволяет системно подойти к формированию алгоритмического мышления у начинающих программистов. Последовательное изучение алгоритмов обработки массивов создает прочный фундамент для освоения более сложных тем программирования. Ключевым преимуществом подхода является его практико-ориентированность: каждый теоретический концепт немедленно подкрепляется решением конкретных задач, что способствует глубокому усвоению материала и развитию навыков, необходимых для реальной разработки программного обеспечения.

Важнейшим результатом применения данной методики является переход студентов от механического запоминания синтаксиса к осознанному применению алгоритмических конструкций, что составляет суть профессионального роста программиста.

Литература

1. Павловская Т.А. C#. Программирование на языке высокого уровня. – СПб.: Питер, 2009. – 432 с.

2. Информационно-справочная система ресурса Microsoft. URL: https://msdn.microsoft.com/uk-ua/library/ms173121(v=vs.90).aspx (дата обращения: 08.10.2025).

3. Ресурс METANIT.COM. URL: https://metanit.com/sharp/tutorial/ (дата обращения: 15.10.2025).

4. Ресурс internetурок. URL: https://interneturok.ru/lesson/algebra/podgotovka-k-ege/tema-2-uravneniya-i-neravenstva/kvadratnye-uravneniya-

lineynye
-
uravneniya
-
praktika
(дата обращения: 16.10.2025).