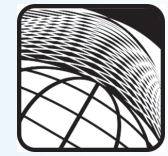

DOI: 10.32517/2221-1993-2024-23-5-5-11

**К. В. Бутарев**
***Московский педагогический государственный университет, г. Москва, Россия***

# ИЗУЧЕНИЕ ПРОГРАММИРОВАНИЯ С ПОЗИЦИИ ТЕОРИИ КОГНИТИВНОЙ НАГРУЗКИ

***Аннотация***

В статье обсуждаются приложения теории когнитивной нагрузки к конструированию задач и образовательных материалов по программированию. Описывается теория когнитивной нагрузки как теория педагогического дизайна, ее основания в когнитивной психологии, а именно трехкомпонентная модель Аткинсона—Шиффрина и модель рабочей памяти Бэддели. Даны определения когнитивной нагрузки и ее категорий, некоторых других важных понятий. Рассматриваются основные когнитивные эффекты, описанные теорией когнитивной нагрузки: эффект разделения внимания, эффект модальности, эффект коллективной рабочей памяти, эффект разобранного примера. Приведены примеры их практического применения в конструировании образовательных материалов по программированию, которые позволяют добиться снижения когнитивной нагрузки при обучении программированию с помощью визуализации, комментариев/комментирования и парного программирования. Уделяется внимание задаче Парсона, которая позволяет снизить внешнюю когнитивную нагрузку и сконцентрировать внимание ученика на важных для обучения аспектах задачи.

***Ключевые слова:*** педагогический дизайн, теория когнитивной нагрузки, когнитивная нагрузка, парное программирование, задача Парсона.

## 1. Введение

В связи с глобальными изменениями, происходящими в области экономики, политики, культуры и общества в целом, можно предполагать, что «будущие поколения <...> будут продолжать формироваться под влиянием дальнейшего развития интернета, соцсетей, компьютеров, смартфонов, массового потребления продуктов и услуг, а также других аналогичных факторов» [6]. Немалую роль в этих изменениях играет цифровизация, которая затрагивает все сферы жизни человека, — она является катализатором стремительного возрастания объемов информации, динамичного изменения структур социальных институтов, в том числе системы образования. Если не затрагивать управленческую сторону данных изменений, а сосредоточиться на изменяющихся ролях учебных предметов, то можно обратить внимание на особую роль информатики. Как замечает Л. Л. Босова, «основные цели отечественного курса школьной информатики <...> [могут быть сведены к] <...> необходимости формирования набора тех знаний и умений, которые нужны человеку для полноценной жизни в современном информационном обществе» [1].

В курсе информатики можно выделить четыре основных укрупненных раздела: «Цифровая грамотность», «Теоретические основы информатики», «Алгоритмы и программирование», «Информационные технологии». Не умаляя потенциал других разделов, заметим, что подготовка по темам такого практико-ориентированного раздела, как «Алгоритмы и программирование», не только позволяет ориентировать учеников на выбор профессии в области информационных технологий,

***Контактная информация***

**Бутарев Кирилл Викторович**, аспирант кафедры теории и методики обучения математике и информатике, Институт математики и информатики, Московский педагогический государственный университет, г. Москва, Россия; *адрес:* 119991, Россия, г. Москва, ул. Малая Пироговская, д. 1, стр. 1; *e-mail:* k1306969@gmail.com

**K. V. Butarev**
Moscow State Pedagogical University, Moscow, Russia

## LEARNING PROGRAMMING FROM THE PERSPECTIVE OF COGNITIVE LOAD THEORY

***Abstract***

The article discusses the applications of Cognitive Load Theory in the design of tasks and educational materials for programming. It describes Cognitive Load Theory as a theory of instructional design, grounded in cognitive psychology, specifically the Atkinson—Shiffrin three-component model and Baddeley's working memory model. The article also defines cognitive load and its categories, as well as other important concepts. Key cognitive effects described by Cognitive Load Theory, such as the split-attention effect, modality effect, collective working memory effect, and worked-example effect, are examined. Practical examples of their application in the design of educational materials for programming are provided, demonstrating how they help reduce cognitive load through the use of visualization, commenting, and pair programming. Special attention is given to Parson's problems, which help reduce extraneous cognitive load and focus the student's attention on the key instructional aspects of the task.

***Keywords:*** instructional design, cognitive load theory, cognitive load, pair programming, Parson's problem.

но и способствует развитию у них важных метапредметных навыков декомпозиции и планирования, анализа и оценки, обеспечивая немалый вклад в становление определенного стиля мышления, все чаще называемого вычислительным, формирование которого Л. Л. Босова предлагает рассматривать «как стратегическую цель общего образования в области информатики и информационных технологий» [1]. При этом само программирование, как один из инструментов формирования вычислительного мышления, является сложным когнитивным навыком, который требует освоения множества компетенций и считается трудным для изучения. С учетом описанных условий можно говорить, что вопрос пересмотра и модернизации подходов к обучению информатике в части изучения основ алгоритмизации и программирования имеет действительно важное значение, а одним из возможных инструментов при этом может стать *педагогический дизайн*.

А. Ю. Уваров определяет педагогический дизайн как «систематическое использование знаний об эффективной учебной работе в процессе проектирования, разработки, оценки и использования учебных и методических материалов» [5]. Коллектив авторов из Института образования НИУ ВШЭ рассматривает педагогический дизайн как «систему процедур по разработке способов доставки учебного содержания (учебных продуктов) учащимся, создаваемую с целью помочь им развить у себя требуемые компетенции» [3]. Теория педагогического дизайна представляет собой интегративный подход, сочетающий в себе как выверенную форму передачи учебной информации, так и ее структурированное представление, которое основывается на базе психолого-педагогических исследований. В рамках данной работы в качестве теоретической основы педагогического дизайна будет рассмотрена теория когнитивной нагрузки, наследующая идеи когнитивной психологии.

## 2. Основные положения теории когнитивной нагрузки

**Теория когнитивной нагрузки** изучает влияние когнитивной нагрузки на процесс обучения, ее создателем считается австралийский психолог Джон Свеллер (John Sweller). Основываясь на идеях когнитивной психологии, данная теория утверждает, что обучающиеся могут эффективно работать с образовательной информацией в том случае, если ее форма и количественные характеристики не перегружают рабочей (кратковременной) памяти.

**Когнитивная нагрузка** — это объем рабочей памяти, который требуется человеку для обработки актуальной информации. Кроме того, стоит заметить, что «когнитивная нагрузка, которую оказывает учебная информация на рабочую память, может быть разделена на категории в зависимости от ее природы»* [15].

Часть нагрузки на рабочую память обусловлена внутренней природой информации, и эта нагрузка называется **внутренней когнитивной нагрузкой**. Ее величина не зависит от используемых методов обучения, — она определяется природой (логической структурой) информации, которую учащийся должен усвоить для достижения образовательных результатов. Часть когнитивной нагрузки, которая зависит от формы представления или вида деятельности, в которых учащимся необходимо участвовать, называется **внешней когнитивной нагрузкой**. Таким образом, общая когнитивная нагрузка на рабочую память определяется внутренней природой изучаемого материала (внутренняя когнитивная нагрузка) и способом его представления (внешняя когнитивная нагрузка)**. Для того чтобы обеспечить наилучшие условия для усвоения учебного материала учеником, необходимо минимизировать внешнюю когнитивную нагрузку. Это должно повлечь за собой освобождение ресурсов рабочей памяти, которые могут быть распределены на работу с внутренней природой изучаемой информации.

Уровень когнитивной нагрузки (как внутренней, так и внешней) зависит от интерактивности элементов. **Интерактивные элементы** — это фрагменты информации, которые должны обрабатываться одновременно в рабочей памяти, так как связаны логической связью, без которой они теряют смысл [15]. Интерактивность определяется не только внутренней природой информации, но также формой, которую задает образовательным материалам дизайнер, например, с помощью добавления лишних логических связей.

Элементы, как правило, представляют собой **схемы** — когнитивные конструкции, позволяющие человеку группировать множество элементов информации в один элемент в соответствии со способом их использования [15], т. е. большинство схем состоит из подсхем или подэлементов. Новая схема образуется в тот момент, когда логические связи между интерактивными элементами усвоены субъектом, после чего данную схему можно рассматривать как отдельный элемент внутри рабочей памяти. Информация в долговременной памяти остается неактивной до тех пор, пока она не будет активирована сигналами из окружающей среды, которые побуждают рабочую память выбрать необходимую *схему*. Таким образом, при изучении нового материала сначала возникает возрастание когнитивной нагрузки за счет перемещения элементов из внешней среды в рабочую память. После чего из долговременной памяти в рабочую перемещается необходимая схема (рис. 1), которая будет изменена за счет включения в себя новых подэлементов.

Мы должны обратить внимание на то, что на рисунке 1 рабочая память изображена в виде столбца с семью слотами, — данная схема построена на основе *«магического числа семь плюс-минус два»* [12] и является в некоторой степени упрощением описания модели рабочей памяти. Джордж Миллер (George Miller), один из основателей когнитивной психологии, показал на

* Ориг. англ.: «The load imposed on working memory by that instructional information can be divided into categories depending on its function».

** Мы не упоминаем о *релевантной* [или уместной] *когнитивной нагрузке* (*англ.* germane cognitive load) в связи с тем, что опираемся на *новую теорию когнитивной нагрузки*. При формулировке новой теории Джон Свеллер исключил данную категорию в связи с тем, что она *«не навязывается»* учебным материалом в отличие от внутренней и внешней категорий когнитивной нагрузки, а также не поддается эмпирическим критериям верификации и фальсификации [15].

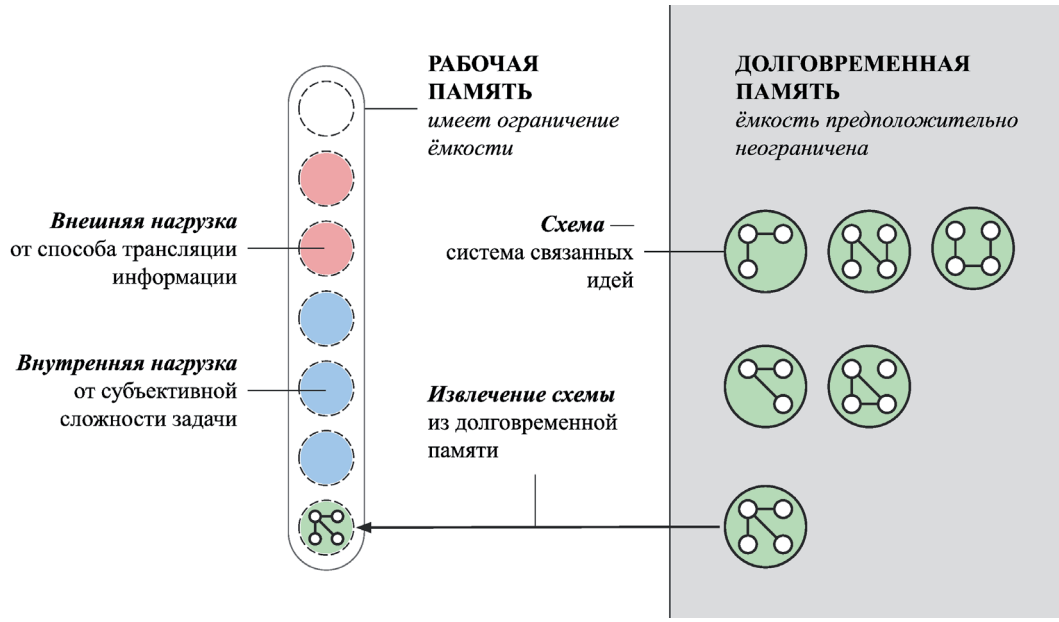


*Рис. 1. Рабочая память (теория когнитивной нагрузки)*

основе ряда исследований, что объем рабочей памяти у человека ограничен примерно семью элементами (в диапазоне от пяти до девяти). Это наблюдение ограниченности ресурса рабочей памяти стало фундаментальным в когнитивных науках.

Теорией когнитивной нагрузки наследуется широко используемая в когнитивной психологии **трехкомпонентная модель памяти Аткинсона—Шиффрина** [7], которая выделяет *сенсорную память*[*], *рабочую память* и *долговременную память*. При этом архитектура рабочей памяти уточняется Свеллером с некоторыми оговорками [15], — опираясь на **модель рабочей памяти Бэддели**[**] **[8]**, автор теории когнитивной нагрузки подвергает логическому сомнению нахождение *центрального управляющего элемента* в структуре рабочей памяти[***], но утверждает со ссылками на ряд исследований, что гипотеза выделения в рабочей памяти отдельных *аудиального* (фонологический цикл) и *визуального* (визуально-пространственный набросок) компонентов, отвечающих за обработку звуковых и визуальных образов соответственно, имеет значительную эмпирическую поддержку. При этом утверждается, что аудиальный и визуальный компоненты (частично) независимы и обладают своими собственными ресурсами памяти.

Модель когнитивной архитектуры человека, используемая в теории когнитивной нагрузки, а также само понятие когнитивной нагрузки позволили исследователям в рамках развития теории определить, спрогнозировать и выявить некоторые закономерности, которые потенциально могут использоваться в качестве принципов для построения образовательных материалов.

## 3. Приложения теории когнитивной нагрузки в обучении программированию

### 3.1. Эффект разделения внимания

**Эффект разделения внимания** (*англ.* split-attention effect) заключается в том, что если обучающийся, стремясь понять и усвоить материал, распределяет внимание между несколькими источниками, то это увеличивает внешнюю когнитивную нагрузку. Такая ситуация возникает, например, если текст и соответствующая ему схема или иллюстрация расположены отдельно друг от друга и обучающемуся необходимо переключаться между ними для интеграции информации. При этом рабочая память вынуждена одновременно обрабатывать и связывать несколько разрозненных фрагментов информации, что может снизить эффективность обучения. Для снижения этого эффекта рекомендуется интегрировать информацию — например, размещать поясняющий текст непосредственно рядом с изображениями или схемами.

При обучении программированию во избежание данного эффекта при конструировании учебных материалов целесообразно придерживаться следующих рекомендаций:

- комбинация диаграмм с подписями и пояснениями;
- аннотация программ с помощью комментариев, выделение общих разделов и шаблонов.

---

[*] *Сенсорная память* — кратковременное хранилище сенсорной информации (например, зрительной или слуховой) на очень короткий период (от долей секунды до нескольких секунд). Если информация не попадает в фокус внимания субъекта, она быстро исчезает.

[**] Модель рабочей памяти была предложена в 1974 году Аланом Бэддели (Alan Baddeley) совместно с Грэмом Хитчем (Graham Hitch). Первоначальная версия включала три компонента: *центральный управляющий элемент*, *фонологический цикл* и *визуально-пространственный набросок*. Позднее, в 2000 году, Бэддели добавил в модель четвертый компонент — *эпизодический буфер*, который был введен для объяснения интеграции информации и взаимодействия между рабочей и долговременной памятью.

[***] Сомнения Свеллера и его коллег касаются логического противоречия, которое заключается в том, что, если центральный управляющий элемент выделен в структуру рабочей памяти, у него нет возможности обращения напрямую к информации, хранящейся в долговременной памяти.

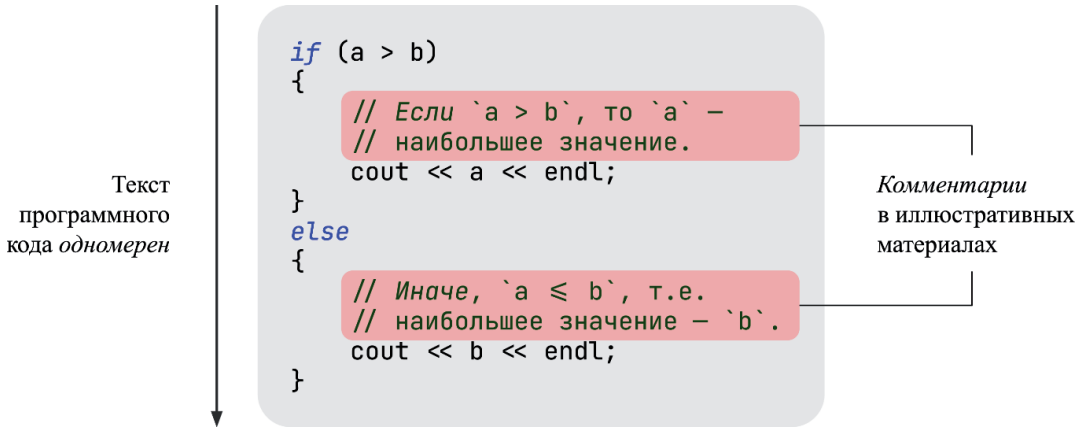


*Рис. 2. Комментарии к программному коду*

На рисунке 2 представлен фрагмент программного кода для решения задачи нахождения максимума из двух значений, использование которого предполагается в качестве иллюстративного материала при изучении темы «Условный оператор». Описание тел оператора IF и оператора ELSE сопровождается комментариями, поясняющими, в каком случае они будут выполнены. Согласно эффекту разделения внимания данный подход более предпочтителен в сравнении с вынесением комментариев в текст вне программного кода.

Рассмотрим эту же задачу с точки зрения эффекта модальности.

### 3.2. Эффект модальности

**Эффект модальности** (*англ.* modality effect) заключается в том, что улучшение усвоения информации происходит тогда, когда она представляется через разные сенсорные каналы — *визуальный* и *аудиальный* — одновременно. Этот эффект основан на том, что рабочая память обрабатывает информацию более эффективно, если нагрузка распределена между двумя каналами восприятия, вместо того чтобы перегружать один канал.

Таким образом, если подобная иллюстрация (рис. 2) нужна не в тексте учебного пособия, а при объяснении материала учителем, то ее можно изменить. Кроме того, стоит обратить внимание на то, что пространство программного кода является одномерным, что накладывает дополнительные ограничения восприятия — при прочтении субъект вынужден переключать свое внимание между программным кодом и комментариями к нему. Как пишет В. В. Давыдов, «переход некоторого объекта в форму модели позволяет обнаружить в нем такие свойства, которые не появляются при непосредственном оперировании» [2]. То есть, когда явление «фиксируется в виде графической схемы, то возможно вычленить в нем его составляющие» [4].

Для того чтобы понизить внешнюю когнитивную нагрузку, можно показать те логические отношения графически, которые существенны для объяснения (эффект разделения внимания), а текстовые комментарии перевести в устную форму. На рисунке 3 изображен тот же самый программный код для решения задачи нахождения максимума из двух значений, только без комментариев. Стрелками на рисунке обозначен поток выполнения в случае истинности или ложности проверяемого условия. Подписи стрелок подразумеваются пояснениями учителя при объяснении конструкции условного оператора.

Эффект модальности может выражаться в *моделировании решения задачи учителем в реальном времени*, подразумевающем детальную демонстрацию процесса решения задачи, который начинается с постановки задачи, формализации условия и декомпозиции задачи до подзадач, а заканчивается реализацией текста программы каждой подзадачи, их синтезом для решения основной задачи и последующим тестированием написанной

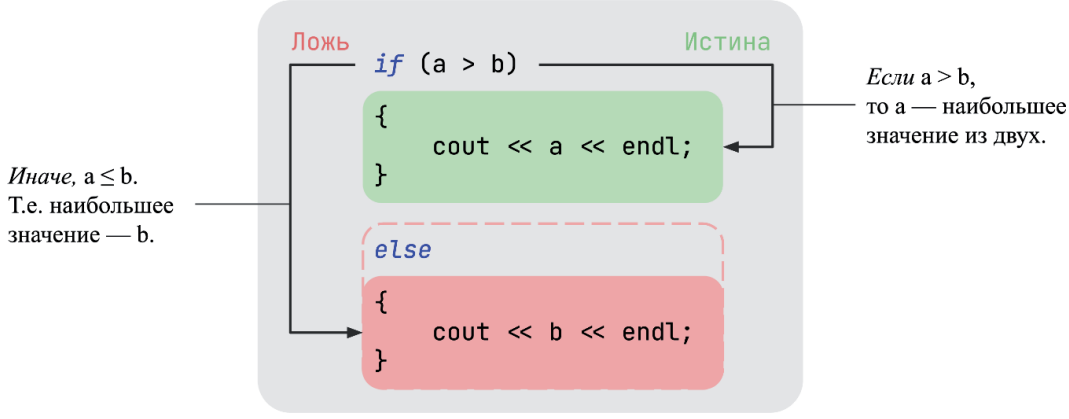


*Рис. 3. Программный код с визуальными и аудиальными пояснениями*

программы [13]. При использовании моделирования решения задачи можно снизить когнитивную нагрузку на рабочую память учеников за счет использования двух каналов — визуального и аудиального. Кроме того, подобный подход может содержать взаимодействие между преподавателем и обучающимися в форме обсуждения решения, рассмотрение альтернативных способов решения и их оценку, что также должно положительно сказаться на результате обучения [11].

### 3.3. Эффект коллективной рабочей памяти

При распределении работы между участниками группы возникает **эффект коллективной рабочей памяти** (*англ.* collective working memory effect): вместо того чтобы каждый обучающийся обрабатывал всю информацию самостоятельно, группа совместно решает задачу, при этом каждый участник отвечает за часть процесса. Это позволяет снизить общую когнитивную нагрузку и улучшить результаты обучения за счет эффективного распределения информации и усилий.

При обучении программированию можно добиться эффекта коллективной рабочей памяти с помощью использования *парного программирования* [13]. Это позволит распределить работу между обучающимися и тем самым снизить когнитивную нагрузку на индивидуальную рабочую память каждого (рис. 4). Однако недостаточная коммуникация между учащимися может привести к *«издержкам»*, которые могут нивелировать преимущества данного эффекта.

**Парное программирование** — это метод разработки программного обеспечения, при котором два программиста работают вместе за одним компьютером, решая одну задачу. В рамках решения задачи в паре обычно выделяются две роли: *роль кодера* и *роль ревизора*. Основная **задача кодера** — разработка алгоритма решения задачи и воплощение его в рамках языка программирования; **задачи ревизора** — отслеживать появление синтаксических и логических ошибок в коде в процессе его написания кодером, искать узкие моменты в условии задачи, которые потенциально могли быть не учтены кодером. Кроме того, члены пары могут обсуждать программный код и условие решаемой задачи.

**Основные когнитивные преимущества парного программирования** [13]:

- вместо того чтобы одному программисту решать всю задачу самостоятельно, работа делится между двумя людьми, что снижает индивидуальную когнитивную нагрузку;
- за счет того что задача поиска ошибок делегируется ревизору, с кодера снимается часть когнитивной нагрузки;
- партнеры обсуждают варианты решения, что помогает лучше понимать сложные концепции и уменьшает необходимость обработки всех элементов задачи в одиночку.

Использование парного программирования помогает снизить внешнюю когнитивную нагрузку на учеников, что происходит за счет разделения задач с разной степенью сложности: задачи, требующие низкого уровня когнитивной нагрузки (например, набор текста, работа с компьютером, чтение условия задачи), отделены от задач, требующих высокого уровня когнитивной нагрузки (например, анализ синтаксиса программного кода, разработка алгоритмов, поиск логических ошибок). Но стоит учитывать, что лучше формировать пары примерно с равными характеристиками владения материалом.

### 3.4. Эффект разобранного примера

**Эффект разобранного примера** (*англ.* worked-example effect) состоит в том, что предоставление обучающимся полностью разобранного по шагам решения задачи помогает им лучше усваивать материал и впоследствии использовать продемонстрированные идеи в решении аналогичных задач. Преподаватель сначала объясняет все этапы решения, отвечая на вопросы учеников при необходимости. Такой подход позволяет обучающимся увидеть, как профессионал или эксперт, в роли которого выступает учитель, решает задачу. Затем преподаватель обращается к обучающимся с вопросами, требуя объяснить ключевые этапы процесса, что стимулирует их к более глубокому осмыслению решения. После этого обычно следует практическая работа, где ученики применяют полученные знания и навыки, отрабатывая их на новых задачах. Свеллер утверждает, что использование

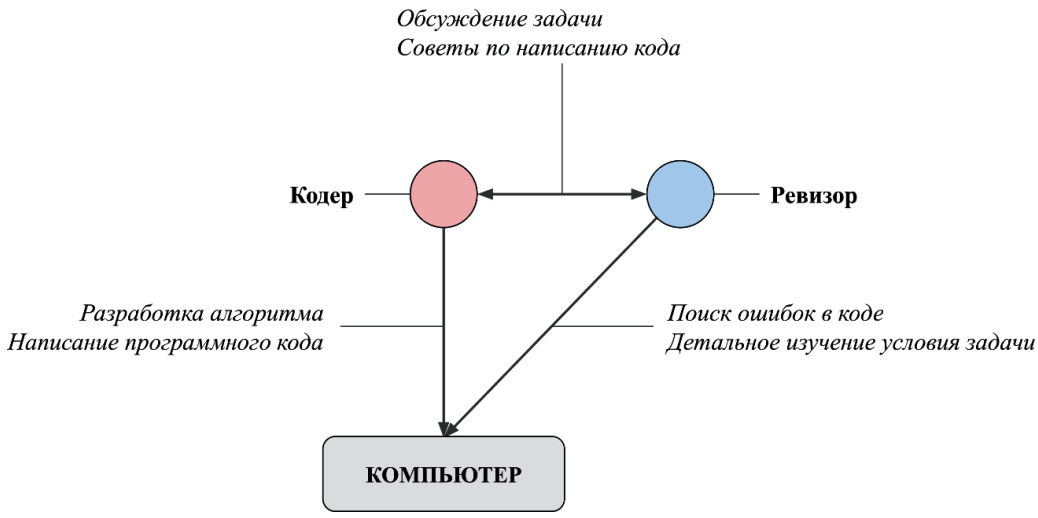


*Рис. 4. Схема разделения задач при парном программировании*

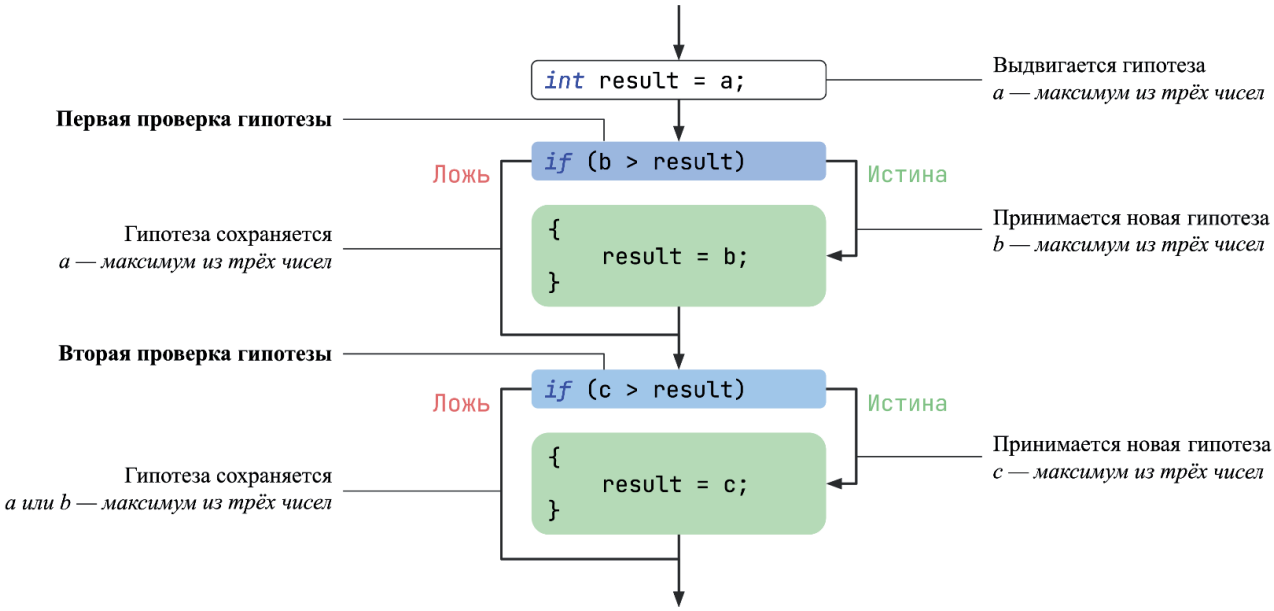


*Рис. 5. Разобранный пример*

разобранного примера позволяет более явно показать логические связи между интерактивными элементами, задействованными в решении задачи (тем самым снижая когнитивную нагрузку), — предоставить готовую схему для сохранения ее в долговременной памяти [15].

В качестве иллюстрации разобранного примера можно привести схему решения задачи о нахождении максимума из трех целых чисел (рис. 5), которая может использоваться в качестве иллюстрации при объяснении решения задачи.

Б. Скуддер (B. Skudder) и Э. Лакстон-Рейли (A. Luxton-Reilly) приводят несколько способов использования разобранного примера [14]:

- *Использование только примеров*. Учащимся предоставляется набор разобранных примеров. С примерами не связаны какие-либо действия, такие как упражнения или задачи для решения.
- *Блоки примеров-задач*. Учащимся предоставляется для изучения блок разобранных примеров различных типов, затем дается набор связанных задач, которые, как ожидается, учащиеся должны решить.
- *Пары примеров-задач*. К каждому разобранному примеру прилагается задача, аналогичная разобранной. Учащиеся поочередно изучают отработанный пример и решают связанную с ним задачу.
- *Разобранные примеры в незаконченной форме*. Учащимся представлен разобранный пример, затем представлен другой разобранный пример с пропущенным шагом, и ожидается, что учащиеся заполнят пропущенный шаг. Таким образом строится серия разобранных примеров, при этом каждый раз удаляется очередной шаг, пока перед учащимися не останется только задача, которую нужно решить.

### 3.5. Задача Парсона

Кроме разобранных выше *эффектов*, которые объясняются теорией когнитивной нагрузки, стоит уделить внимание конкретной форме задачи, которая, как показал ряд исследований [9, 10], действительно уменьшает когнитивную нагрузку в сравнении с классическим решением задач по программированию, во всяком случае, у новичков, — *задаче Парсона*.

**Задача Парсона** — это форма задачи, используемая при обучении программированию, в которой учащимся предлагается готовый код, но в виде отдельных фрагментов, которые необходимо правильно упорядочить для получения рабочей программы (рис. 6). Основная цель этой задачи — помочь обучающимся развить навыки понимания структуры программного кода и его логики без необходимости писать код с нуля.

В связи с тем, что при решении задач, поставленных в такой форме, у учеников нет необходимости в написании программного кода средствами языка программирования, синтаксические правила которого обычно не связаны с самой логикой вычислений, утверждается, что снижается внешняя когнитивная нагрузка в сравнении с классическим решением задачи. Кроме того, при правильном конструировании заданий по решению сложных задач можно также снизить и внутреннюю когнитивную нагрузку — для этого можно разделить крупную задачу на несколько более простых, которые ученику будет необходимо решить отдельно.

В целом задача Парсона позволяет снижать в первую очередь внешнюю нагрузку и фокусировать внимание на обучающих аспектах задачи, опуская особенности языка программирования.

## 4. Заключение

В данной статье была рассмотрена теория когнитивной нагрузки и ее значение в контексте разработки образовательных материалов для обучения программированию. Рассматриваемые в статье эффекты, описанные теорией когнитивной нагрузки, — эффект разделения внимания, эффект модальности, эффект коллективной рабочей памяти и эффект разобранного примера — были

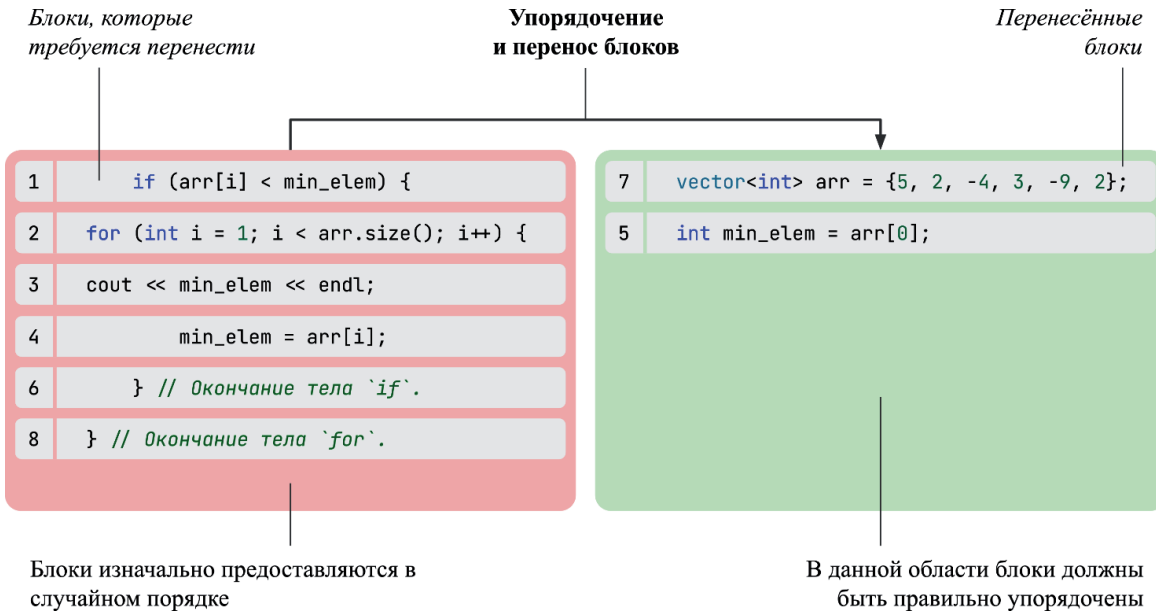


*Рис. 6. Задача Парсона*

проиллюстрированы задачами по программированию для создания эффективных методик обучения. Предполагается, что конструирование образовательных материалов на основе положений теории когнитивной нагрузки может повысить эффективность обучения программированию.

## Список источников

1. *Босова Л. Л.* Вычислительное мышление как стратегическая цель общего образования в области информатики и информационных технологий // Актуальные вопросы методики обучения информатике в условиях цифровой трансформации образования. М.: МПГУ, 2024. С. 24–33. EDN: AGFBPC.

2. *Давыдов В. В.* Виды обобщения в обучении: Логико-психологические проблемы построения учебных предметов. М.: Педагогическое общество России, 2000. 480 с.

3. Педагогический дизайн: российская и зарубежная исследовательская повестка / Е. В. Чернобай (научная редакция), Е. А. Ефимова, Ю. Н. Корешникова, М. А. Давлатова; НИУ ВШЭ, Институт образования. М.: НИУ ВШЭ, 2022. 44 с. (Современная аналитика образования. № 3 (63).)

4. *Салмина Н. Г.* Знак и символ в обучении. М.: Изд-во Московского университета, 1988. 288 с.

5. *Уваров А. Ю.* Педагогический дизайн // Информатика. 2003. № 3. С. 1–32.

6. *Урсул А. Д.* Цифровизация и образование для устойчивого развития: перспективы взаимосвязи в процессе эволюции // Знание. Понимание. Умение. 2020. № 2. С. 39–54. EDN: CZQPZN. DOI: 10.17805/zpu.2020.2.4.

7. *Atkinson R. C., Shiffrin R. M.* Human memory: A proposed system and its control processes. // The Psychology of Learning and Motivation / J. T. Spence & K.W. Spence (Eds.). 1968. P. 89–196.

8. *Baddeley A.* Working memory: Theories, models, and controversies // Annual Review of Psychology. 2012. Vol. 63. P. 1–29. DOI: 10.1146/annurev-psych-120710-100422/

9. *Haynes C. C., Ericson B. J.* Problem-solving efficiency and cognitive load for adaptive parsons problems vs. Writing the equivalent code // Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. 2021. P. 1–15. DOI: 10.1145/3411764.3445292.

10. *Hou X., Ericson B. J., Wang X.* Understanding the effects of using parsons problems to scaffold code writing for students with varying CS self-efficacy levels // Proceedings of the 23rd Koli Calling International Conference on Computing Education Research. 2024. New York: Association for Computing Machinery. P. 1–12. DOI: 10.1145/3631802.3631832.

11. *Linn M. C., Clancy M.* The case for case studies of programming problems // Communications of the ACM. 1992. Vol. 35. Is. 3. P. 121–132. DOI: 10.1145/131295.131301.

12. *Miller G. A.* The magical number seven, plus or minus two: Some limits on our capacity for processing information // Psychological Review. 1956. Vol. 63. P. 81–97.

13. *Sands P.* Addressing cognitive load in the computer science classroom // ACM Inroads. 2019. № 1 (10). P. 44–51. DOI: 10.1145/3210577.

14. *Skudder B., Luxton-Reilly A.* Worked examples in computer science // Proceedings of the Sixteenth Australasian Computing Education Conference. 2014. Auckland: Australian Computer Society. P. 59–64.

15. *Sweller J., Ayres P., Kalyuga S.* Cognitive Load Theory. New York: Springer Science+Business Media, 2011. 274 p.