МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им.В.П.АСТАФЬЕВА
(КГПУ им.В.П.Астафьева)

Институт/факультет Математики, физики и информатики

Выпускающая кафедра Базовая кафедра информатики и информационных технологий в образовании

**Марченко Любовь Сергеевна**

# МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Тема Обучение объектно-ориентированному программированию в педагогическом ВУЗе на основе ментально- телесного подхода

Направление подготовки 44.04.01 Педагогическое образование

Профиль IT-технологии в образовании

**ДОПУСКАЮ К ЗАЩИТЕ**

Заведующий кафедрой
д.п.н., профессор Пак Н.И.
(дата, подпись)

Руководитель магистерской программы
д.п.н., профессор Пак Н.И.
(дата, подпись)

Научный руководитель
к.п.н. доцент Степанова Т. А.
(дата, подпись)

Дата защиты

Обучающийся Марченко Л.С.
(фамилия, инициалы)
(дата, подпись)

Оценка

**Красноярск 2018**

## Реферат

Магистерская диссертация состоит из 84 страниц, 20 рисунков, 40 источников литературы и 5 приложений

**Объект исследования:** Процесс обучения объектно-ориентированному программированию

**Предмет исследования:** Обучение объектно-ориентированному программированию на основе ментально-телесного подхода

**Цель исследования:** Теоретически обосновать возможность и необходимость ментально-телесного подхода к обучению объектно-ориентированному программированию, определить и разработать необходимые для его реализации средства обучения.

**Задачи исследования:**

1) Рассмотреть теоретические основы ментально-телесного подхода к обучению программированию.
2) Выявить особенности обучения объектно-ориентированному программированию в педвузе.
3) Уточнить понятие объектного стиля мышления, построить его структурную модель, рассмотреть средства и методы его развития и определить способы диагностики.
4) Разработать ментальные карты и кинестетические тренажеры по теме «Классы в C++» курса «Языки и методы программирования», и провести апробацию разработанных материалов

В рамках магистерской диссертации были использованы следующие **методы исследования:** теоретические (изучение и анализ педагогической, психологической, методической и предметной литературы по теме исследования, анализ теоретических и эмпирических данных, изучение и обобщение педагогического опыта, сравнительный анализ, классификация);

эмпирические (наблюдение, анкетирование, беседа, педагогический эксперимент).

**Научная новизна исследования** состоит в том, что:

1. обоснована возможность и необходимость использования ментально-телесного подхода к обучению ООП в педвузе
2. уточнено понятие объектный стиль мышления (ОСМ), построена его структурная модель, определены уровни сформированности
3. предложена диагностика уровня сформированности ОСМ
4. определены средства развития ОСМ – ментальные карты, кинестетические тренажеры
5. спроектированы принципиально новые средства обучения ООП – кинестетический тренажер по теме «Классы в С++»

**Теоретическая значимость исследования** заключается уточнении понятия объектный стиль мышления (ОСМ) и построении его структурной модели

**Практическая значимость исследования** заключается в том, что:

1. Предложена диагностика уровня сформированности ОСМ
2. Разработаны ментальные карты по теме «Классы в С++»»
3. Спроектирован кинестетический тренажер по теме «Классы в С++»

Результаты магистерского исследования апробированы в ИМФИ КГПУ в рамках курса «Профильное исследование» и могут быть использованы при обучении теме «Классы в С++» курса «Языки и методы программирования»

## Abstract

The master's thesis consists of 84 pages, 20 figures, 40 sources of literature and 5 applications

**Object of study:** The process of learning object-oriented programming

**Object of the study:** Training in object-oriented programming based on the mental-corporeal approach

**Purpose of the research:** To theoretically substantiate the possibility and necessity of a mentally-bodily approach to learning object-oriented programming, to determine and develop the training tools necessary for its implementation.

**Objectives of the study:**

1) Consider the theoretical foundations of the mental-bodily approach to teaching programming.

2) Identify the features of learning object-oriented programming in the pedagogical college.

3) Clarify the concept of the object style of thinking, build its structural model, consider the means and methods of its development and determine the methods of diagnosis.

4) Develop mental maps and kinesthetic simulators on the subject "Classes in C ++" in the course "Languages and methods of programming", and conduct approbation of the developed materials

Within the framework of the master's thesis the following research methods were used: theoretical (study and analysis of pedagogical, psychological, methodological and subject literature on the research topic, analysis of theoretical and empirical data, study and generalization of pedagogical experience, comparative analysis, classification); Empirical (observation, questioning, conversation, pedagogical experiment).

**The scientific novelty of the study is that:**

1. The possibility and necessity of using a mentally-bodily approach to teaching PLO in the pedagogical college

2. The concept of the object style of thinking (OSM) is clarified, its structural model is constructed, levels of formation are determined

3. Diagnosis of the OSM generation level

4. The means of development of the OSM - mental maps, kinesthetic simulators

5. fundamentally new tools for teaching OOP are designed - a kinesthetic simulator on the topic "Classes in C ++"

The theoretical significance of the research is to clarify the concept of the object style of thinking (OCM) and the construction of its structural model

The practical significance of the study is that:

1. The diagnostics of the OSM formation level

2. Mental maps on the topic "Classes in C ++" have been developed "

3. A kinesthetic simulator on the subject "Classes in C ++" was designed

The results of the master's study are tested in the IMPI of the KSPU within the framework of the "Profile research" course and can be used in teaching the topic "Classes in C ++" in the course "Programming languages and methods"

**Оглавление**

ВВЕДЕНИЕ 7
ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МЕНТАЛЬНО-ТЕЛЕСНОГО ПОДХОДА К ОБУЧЕНИЮ ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ 12
1.1 Сущность ментально-телесного подходов к обучению программированию 12
1.2 Особенности изучения ООП 32
1.3. Уточнение понятия объектного стиля мышления. 51
Выводы по 1 главе 53
ГЛАВА 2. УСЛОВИЯ РЕАЛИЗАЦИИ МЕНТАЛЬНО-ТЕЛЕСНОГО ПОДХОДА К ОБУЧЕНИЮ ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ 54
2.1 Диагностика уровня сформированности объектного стиля мышления. 54
2.2 Разработка ментальных карт по теме «Классы в С++» курса «Языки и методы программирования» 60
2.3 Разработка кинестетического тренажера по теме «Классы в С++» курса «Языки и методы программирования» 65
2.4 Результаты апробации 70
Выводы по 2 главе 74
ЗАКЛЮЧЕНИЕ 75
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 78
ПРИЛОЖЕНИЯ 1 83
Приложение 1. 83
Структурная модель объектного стиля мышления 83
Приложение 2. 84
Алгоритмическая ментальная карта «Телефонная книга» по теме «Классы в С++» курса «Языки и методы программирования» 84
Приложение 3. 85
Алгоритмическая ментальная карта «Учет успеваемости студентов» по теме «Классы в С++» «Языки и методы программирования» 85
Приложение 4. 86
Материалы для диагностики уровня сформированности объектного стиля мышления 86
Приложение 5. 88
Кинестетический тренажер по теме «Классы в С++». 88

# ВВЕДЕНИЕ

Современные образовательные стандарты предъявляют высокие требования к предметной подготовки учителя. Учитель информатики должен владеть всеми современными технологиями программирования, поэтому в курс «Языки и методы программирования» в педагогическом вузе включено изучение не только императивного программирования, которое изучается в школе, но и логического, функционального, объектно-ориентированного, параллельного и др. современных технологий программирования.

Изучение объектно-ориентированного программирования вызывает определенные сложности у студентов в силу своих особенностей. Поэтому поиск новых подходов к разработке методических систем обучения объектно-ориентированному программированию (ООП) является актуальной проблемой.

Ментальный подход к обучению предполагает смещение целеполагания учебного процесса в сторону развития когнитивных способностей обучаемых, в частности, если рассматривать обучение программированию – то основной целью будет являться развитие алгоритмического стиля мышления. В рамках ментального подхода к обучению программированию предполагается использование методики ментальных карт как средства развития алгоритмического мышления.

Согласно положениям такого относительно нового течения в психологии как телесный подход, немаловажную роль в формировании мышления вообще, и, алгоритмического мышления в частности играют тактильные, моторные ощущения, кинестетические каналы восприятия.

Однако в настоящее время практически не существует средств обучения, ориентированных на кинестетические каналы восприятия.

Актуальность и практический аспект данной проблемы связаны с необходимостью поиска новых эффективных подходов к обучению. Одним из перспективных видится ментально-телесный подход.

Анализ современного состояния проблемы позволил выделить следующие **противоречия:**

- между требованиями ФГОС к уровню предметной подготовки будущих учителей информатики и недостаточным количеством средств и методов обучения современным парадигмам и технологиям программирования;

- между наличием методических систем обучения студентов объектно-ориентированному программированию и слабой проработкой вопросов их использования с учетом когнитивных способностей обучающихся;

- между возможностями ментальных и телесных технологий обучения и недостаточной соответствующей методической базой.

И обозначить **проблему исследования:**

Какие средства и методы обучения использовать при обучении объектно-ориентированному программированию в педвузе, чтобы сформировать необходимый для будущего учителя информатики уровень объектного стиля мышления, способствующий высокому качеству обучения объектно-ориентированному программированию

**Объект:** процесс обучения объектно-ориентированному программированию.

**Предмет:** обучение объектно-ориентированному программированию на основе ментально-телесного подхода.

**Цель:** на основе ментально-телесного подхода разработать средства обучения объектно-ориентированному программированию, способствующие развитию объектного мышления

**Гипотеза**:

Сформировать необходимый уровень алгоритмического мышления будущих учителей информатики, обеспечивающий высокое качество обучения объектно-ориентированному программированию возможно, если:

- будет уточнено понятие объектного стиля мышления, предложены способы его диагностики;
- обучение будет происходить с опорой на когнитивные особенности познания, учитывая процессуальную структуру мыслительных операций, основанную на информационной модели памяти;
- использовать средства обучения ООП, отражающие мыслительный процесс решения задачи, нацеленные не только на аудиальные и визуальные, но и на кинестетические каналы восприятии - ментальные карты и кинестетические тренажеры.

В ходе исследования были поставлены следующие **задачи**:

1) Рассмотреть теоретические основы ментально-телесного подхода к обучению программированию.
2) Выявить особенности обучения объектно-ориентированному программированию в педвузе.
3) Уточнить понятие объектного стиля мышления, построить его структурную модель, рассмотреть средства и методы его развития и определить способы диагностики.
4) Разработать ментальные карты и кинестетические тренажеры по теме «Классы в C++» курса «Языки и методы программирования», и провести апробацию разработанных материалов

Методологическим обоснованием исследования явились работы исследователей в области психологии А.Н.Леонтьева, С.Л.Рубинштейна**,** У.Найсера, Р.Амтхауера, А.Л.Алюшина, Е.Н.Князевой, А.Ш. Тхостова, Р. Бир, Р. Брукс, Т. ван Гелдер, Э. Кларк, Ж. Лакофф, П. Маес, Э. Прем, Э. Телен, Ф. Варела; методики обучения программированию

А.П.Газейкиной, П.Б.Хорева, Г.С.Иванова, Н.И.Пака, Т.А.Степановой; в области теоретического программирования Николауса Вирта, Грэди Буча, Алана Кея, Бьерна Страуструпа, в области методики ментальных карт Тони Бюзана и Жозефа Новака.

**Научная новизна исследования** состоит в том, что:

1. обоснована возможность и необходимость использования ментально-телесного подхода к обучению ООП в педвузе

2. уточнено понятие объектный стиль мышления (ОСМ), построена его структурная модель, определены уровни сформированности

3. предложена диагностика уровня сформированности ОСМ

4. определены средства развития ОСМ – ментальные карты, кинестетические тренажеры

5. спроектированы принципиально новые средства обучения ООП – кинестетический тренажер по теме «Классы в С++»

**Теоретическая значимость** настоящей работы заключается в уточнении понятия объектного стиля мышления и построении его структурной модели.

**Практическая значимость** работы состоит в разработке системы тестов для диагностики уровня сформированности объектного стиля мышления и принципиально новых средств обучения ООП - ментальных карт и кинестетического тренажера по теме «Классы в С++»

Результаты исследования были представлены на XVI Международном научно-практическом форуме студентов, аспирантов и молодых ученых «Молодежь и наука XXI века», Красноярск, май 2016 г. ; IV Международном Научно-образовательном форуме. «Человек, семья и общество: история и перспективы развития», Красноярск, декабрь 2015 г.; VII Международной научно-практической конференции «Информация и образование: границы коммуникаций» INFO'15, Горно-Алтайск, 2015 г.; Сборник научных трудов по материалам международной научно-

практической конференции «Вопросы образования и науки: теоретический и методический аспекты» Тамбов, 30 июня, 2015г. Международная научно-практическая конференция «Актуальные вопросы в научной работе и образовательной деятельности» Тамбов, 30 мая 2015 г. Международная научно- практическая конференция «Вопросы образования и науки» Тамбов, декабрь 2015г, XIV Всероссийской (с международным участием) научно-практической ,конференции студентов, аспирантов и молодых ученых, Красноярск, май 2015 г. XXI Международный научно-практический форум студентов, аспирантов и молодых ученых «Актуальные проблемы информатики и информационных технологий в образовании», Красноярск, 23 мая 2017г, Международная научно-практический форум студентов, аспирантов и молодых ученых «молодежь и наука XXl века» 22 мая 2018 г. По результатам исследования имеется 9 публикаций (статей и тезисов выступления).

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МЕНТАЛЬНО-ТЕЛЕСНОГО ПОДХОДА К ОБУЧЕНИЮ ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ

## 1.1 Сущность ментально-телесного подходов к обучению программированию

Ментальный подход к обучению предполагает смещение целеполагания учебного процесса в сторону развития когнитивных способностей обучаемых, в частности, если рассматривать обучение программированию – то основной целью будет являться развитие алгоритмического стиля мышления.

Алгоритмический способ мышления позволяет принимать оптимальные решения в любой сфере человеческой деятельности, и сам по себе никак не взаимодействует с программированием и вычислительной техникой. В таком неком виде он существовал постоянно, то есть изначально принадлежал человеческому мышлению. Благодаря появлению вычислительной техники и профессии программиста привело к тому, что определенный круг специалистов нуждается в необходимости алгоритмического способа мышления. Алгоритмический способ для выполнения программистских задач является единственно возможным. Общий алгоритмический подход в программистской практике углубляется и детализируется: структура предметной области становится формализованной информационной структурой, в ней вычленяются количественные взаимосвязи, образующие математическую модель предметной области, превращение алгоритма в компьютерную программу. Т. е. у людей программистов, алгоритмическое мышление развито на более высоком, профессиональном уровне. Для того, чтобы сформировать алгоритмическое мышление у школьников, студентов на повседневном

уровне, у самого учителя оно должно быть развито на профессиональном уровне. Кроме того, современные подходы к совершенствованию системы среднего образования предъявляют высокие требования к уровню предметной подготовки учителя. Несомненно, его педагогический кругозор должен намного превышать рамки школьной программы. Следовательно, учитель информатики должен владеть всеми современными парадигмами и технологиями программирования

По этой причине вузовские курсы программирования предполагают изучение языков программирования, относящихся к различным парадигмам программирования, сложившимся в современной науке – не только императивной, но и декларативных – функциональной, объектно-ориентированной, логической. Современный уровень развития суперкомпьютерных технологий, основанных на параллельных вычислениях, повсеместное использование многоядерных процессоров предполагает также изучение параллельного программирования.

Изучение языка программирования, относящегося к другой парадигме, вызывает определенный ряд сложностей. Так как при переходе к программированию методами, которые относятся к другой парадигме, необходимо изменить не только подход к решению поставленной задачи, но и перестроить мыслительную деятельность относительно новой парадигмы. Каждая парадигма программирования предполагает формирование определенного стиля мышления – объектного, функционального, логического, параллельного [4].

Поскольку в школьном курсе информатики изучаются языки программирования, относящиеся к императивной парадигме, можно сказать, что у студентов в начале обучения сформировано алгоритмическое мышление в узком смысле, в том понимании, которое сложилось в тот период времени, когда преобладала парадигма императивного, структурного программирования.

Произошедший переход к объектно-ориентированной парадигме создания и использования средств информационных технологий не отрицает необходимости формирования алгоритмического стиля мышления, но расширяет это понятие. На современном этапе развития информатики для успешного взаимодействия с компьютером необходим стиль мышления, который можно назвать *объектным* [1]. Он предполагает умение разделить сложную систему на объекты и выстроить их иерархию, т. е. произвести объектную декомпозицию системы, а затем описать поведение этих объектов. Основной операцией при таком стиле является объектно-ориентированная декомпозиция, разложение объектов. Всевозможные классификации по различным логическим основаниям и логические методы формирования понятий составляют значительную часть методов, используемых при таком стиле мышления.

При описании событий используется алгоритмическая декомпозиция системы и необходим алгоритмический стиль мышления. Выбор способов обработки информации определяется спецификой предметной области решаемой задачи. Использование различных подходов к программированию существенно влияет на эффективность процесса создания компьютерных программ. Так, например, процесс разработки высокоинтеллектуальной экспертной программной системы существенно упрощается при использовании логической парадигмы. В связи с этим выделяется логический стиль мышления. В основе логических языков программирования лежит формализованная логика в отличие от императивных языков программирования, ориентированных на компьютер, т. е. основной принцип состоит в том, что нужно подробно, на логически точном языке описать условие задачи. Решение ее получается в результате определенного процесса, который выполняет компьютером. В этом заключается разница между логическими языками программирования и традиционными, требующими описания процедуры решения задачи.

Изучение логической парадигмы обработки информации дает новое понимание при изучении программирования, способствуя тем самым развитию у студентов иного стиля мышления, предполагающего отказ от императивных стереотипов. Идея функционального программирования опирается на интуитивное понятие о функциях как о достаточно общем механизме представления и анализа решений сложных задач. Механизм функций основательно изучен математиками, и это позволяет программистам наследовать выверенные построения, обладающие предельно высокой моделирующей силой.

Функциональный стиль объединяет разные подходы к определению процессов вычисления на основе достаточно строгих абстрактных понятий и методов символьной обработки данных. Связь функционального программирования с математическими основами позволяет в тексте программы наследовать доказательность построения результата, если она достигнута, причем с использованием разных методов абстрагирования решаемой задачи. Сложность решения задач с помощью функциональных определений преодолевается чисто алгебраически: нацеленностью на формализацию основного множества объектов и определения полной семантической системы операций над ними. Это позволяет представлять классы задач и их решение строгими формулами, для наглядности упрощаемыми введением дополнительных функциональных символов. При необходимости такие символы вносятся в определение алгебраической системы, что приводит к ее расширению.

Вводятся новые функции, подобные леммам и другим вспомогательным построениям в математике. Активно используются рекурсия и символьные обозначения как данных, так и действий любых формул, удобных при определении функций [4].

Технология параллельного программирования не выделяется в отдельную парадигму, т.к. она может быть реализована в каждой из них. В

отличие от программирования последовательных вычислений, концептуальную основу которого составляет понятие алгоритма, реализуемого по шагам строго последовательно во времени, в параллельном программировании программа порождает совокупность параллельно протекающих процессов обработки информации, полностью независимых или связанных между собой статическими или динамическими пространственно-временными, причинно-следственными или информационными отношениями. В процессе изучения параллельной технологии программирования формируется параллельный стиль мышления, предполагающий способность к предварительному умозрительному «распараллеливанию» поставленной задачи – ее анализу с целью выделения подзадач, которые могут выполняться параллельно, «распараллеливанию» потоков данных – выделению потоков данных, которыми будут обмениваться выполняемые параллельно подзадачи. Также такой стиль предполагает возможность удерживания в памяти действий всех подзадач в определённый промежуток времени для правильного управления их совместной работой.

При достаточно высоком уровне развития параллельного стиля мышления программист способен предвидеть те проблемы, возникающие при работе параллельных алгоритмов, которые не встречаются в работе последовательных, проявляют себя не каждый раз и существенно затрудняют отладку программ. Лишь достижение определённого уровня сформированности параллельного стиля мышления позволит программисту эффективно реализовывать параллельные алгоритмы [20].

Таким образом, мы видим, что вузовский курс программирования требует значительного расширения алгоритмического стиля мышления – чтобы не путать с его исторически-традиционным пониманием, назовем его стилем мышления современного программиста (Рис.1).

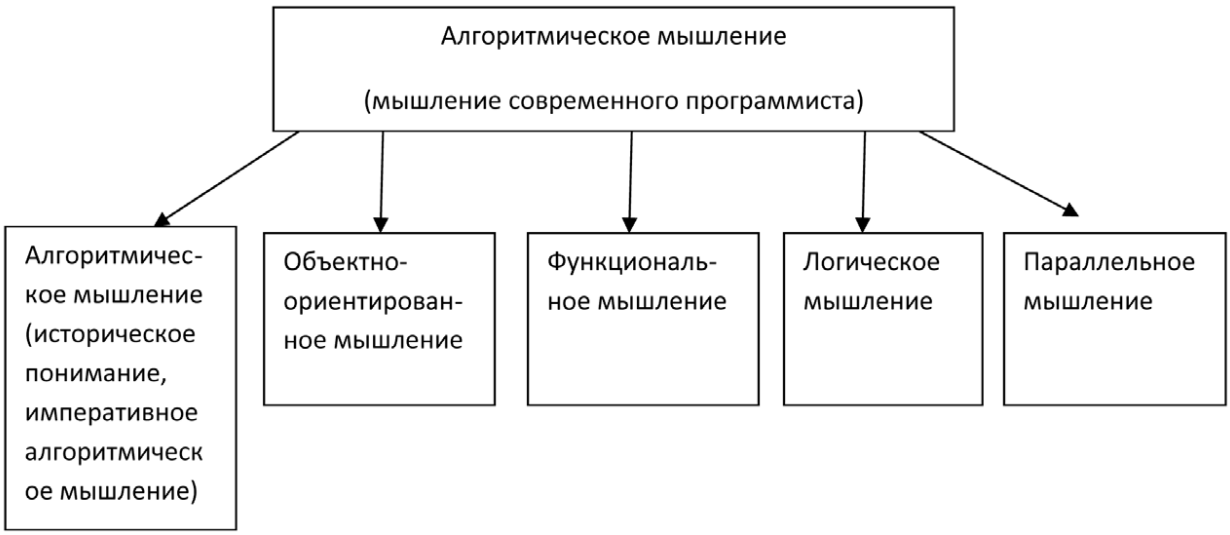



Рис. 1. Расширение понятия «алгоритмическое мышление» (мышление современного программиста)

Если говорить о профессиональном алгоритмическом мышлении будущих учителей информатики, это понятие должно быть расширено еще и методическим компонентом, поскольку у педагогов не только у самих должно быть сформировано алгоритмическое мышление на самом высоком уровне, но они еще должны быть способными формировать и развивать алгоритмическое мышление своих учеников. Обучение программированию в вузе должно быть направлено на формирование у студентов перечисленных выше стилей мышления, без чего это обучение не будет являться эффективным**. [**17]

**Использование ментальных карт в процессе развития алгоритмического мышления**

Опыт изучения программирования и обучения программированию показывает, что основные трудности возникают не при изучении синтаксиса и основных конструкций языка программирования, а на первом этапе решения задач по программированию, на этапе алгоритмизации. И связаны эти трудности с недостаточным уровнем сформированности алгоритмического мышления обучаемых, с их неготовностью воспринимать материал достаточно высокого уровня абстракции и логики. Следовательно,

для разработки методических приемов, позволяющих повысить эффективность обучения и успешность изучения программирования, необходимо обратиться к исследованию процессов мышления изучению мыслительных процессов подходят с разных сторон. Существуют психологические, физиологические и кибернетические теории мышления. В своих исследованиях за основу мы взяли теорию психолога Найссера о том, что мышление человека основывается на ментальных схемах [15].

Наши возможности ориентироваться в пространстве и во времени, осуществлять деятельность говорят о том, что в нашей памяти формируются и хранятся пространственные, временные и деятельностные ментальные схемы. Попытки формализовать процессы мышления, зафиксировать существующие в мозгу человека ментальные схемы привели к понятию ментальной карты как модели ментальной схемы.

Ментальные карты – один из эффективных инструментов организации знаний, концепций и понятий, родившийся на основе психологии познания Дэвида Аусубела. В 60-е годы теорию развил профессор Корнелльского университета Джозеф Новак. Он разработал правила создания ментальных карт – инструмента визуализации и создания (проработки) новых идей или концепций. В основе концепции ментальных карт лежат представления о принципах работы человеческого мозга: ассоциативное (нелинейное) мышление, визуализация мысленных образов, целостное восприятие (гештальт).

Дальнейшее развитие теория получила в работах психолога Тони Бьюзена. В 1974 году он опубликовал книгу «Работай головой», в которой описал метод ментальных карт [2]. Бьюзен превратил метод в коммерческий продукт. Не менее успешно, чем в коммерции, ментальные карты применяются в педагогике. При традиционном изучении учебного материала, как правило, активизируется лишь незначительная часть огромных возможностей мозга. Традиционно тексты учебников, содержание

лекций состоит из фраз, списков и цифр. При его восприятии используются принципы запоминания, рассчитанные на функции левого полушария головного мозга, отвечающего за язык, логику, составление списков и операции с числами, но совсем не учитываются такие аспекты работы мозга, как воображение, ассоциативность, цвет, ритм и ощущения.

В ментальных картах изучаемый материал представляется в виде ключевого образа, воплощающего в себе главную тему. От этого центрального образа отходят соединительные линии, над которыми написаны или нарисованы ключевые понятия для составления образа. Эти линии в свою очередь соединены с другими, на которых расположены ключевые слова, описывающие образы или сами ключевые образы. Таким способом выстраивается многомерная, ассоциативная и образная «карта памяти» (Mind Map) всего материала. Для большей наглядности и лучшего понимания 45 можно изменять размер и стиль используемого шрифта, использовать цвет и рисунки. Метод используется в учебном процессе в следующих вариантах: объяснение темы, подкрепление понимания и запоминания, проверка знаний и выявление неправильно понятого материала.

Современные физиологические теории мышления позволяют сделать вывод о том, что с научных позиций сознание – это инструмент мозга, они определяют биологические функции сознания, предоставляют возможность изучать со- знание объективными методами, выявляют нервные и клеточные основы сознания. Например, в теории когов выявляются биологические, на уровне клеток мозга, ментальные схемы. Согласно положениям этой теории, коги – это системы нейронов, хранящие определенный образ (пространственный, временной или деятельностный). Кибернетика, занимаясь вопросами создания искусственного интеллекта, пытаясь формализовать и автоматизировать процесс мышления, предлагает три

основные модели представления знаний: семантические сети, фреймы, логика.

В исследованиях в области искусственного интеллекта отмечается, что более других соответствует современным представлениям об организации долговременной памяти человека именно семантические сети, однако большинство систем искусственного интеллекта использует фреймы и логику предикатов, поскольку они более формализованы и вследствие этого более эффективно поддаются компьютерной реализации. В подавляющем большинстве работ, посвященных использованию ментальных карт в учебном процессе, ментальные карты представляются в виде простой семантической сети или даже семантического графа, поскольку в них только 46 обозначены понятия как узлы семантической сети и установлены связи между понятиями (нарисованы стрелочки между узлами). Для того, чтобы семантический граф превратился в семантическую сеть, надо указать еще и тип связи (чтобы эти стрелочки были подписаны). А чтобы ментальная карта, представляющая собой модель ментальной схемы, не сводилось к семантической сети, еще необходимо, чтобы в ней был отражены не только понятия и связи между ними, но и некоторые знаки, символы, активизирующие чувственные образы. Тогда она будет работать не только с модельной, понятийной и абстрактной зонами памяти, но и с чувственной зоной памяти. Только в этом случае, как нам кажется, ментальная карта будет действительно ментальной, представлять собой модель ментальной схемы и будет обеспечивать более эффективное восприятие информации и более эффективное протекание процесса мышления.

Для примера рассмотрим существующую у каждого человека ментальную схему времен года. В упрощенной версии, сведенная к семантической сети, она выглядит как схема, представленная на рис. 2.

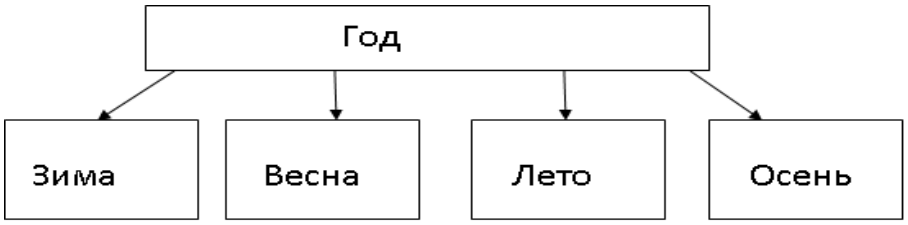



*Рис.2.Времена года. Семантическая сеть*

Если же добавить снежинки, капель, цветы, желтые листья, обозначить цикличность смены времен года – вот тогда эта семантическая сеть действительно превратится в ментальную карту (рис. 3), более соответствующую той ментальной схеме, которая у нас сформирована.

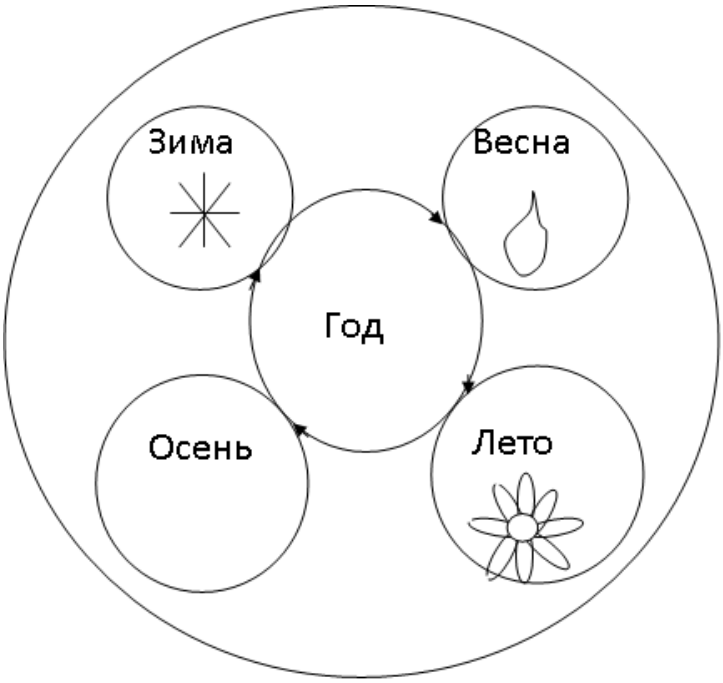



*Рис.3. Времена года. Ментальная карта*

В этой связи можно сделать предположение, что к ставшим классическими моделям представления знаний, современные психологические и биологические теории мышления позволяют добавить еще одну – ментальные карты. Возвращаясь к проблеме использования методики ментальных карт в обучении программированию, можно сделать вывод, что в этом аспекте нас будут интересовать деятельностные ментальные схемы, поскольку ментальной картой таких схем является не что иное, как алгоритм.

Существует классический способ формализации алгоритма, записанного на естественном языке в виде блок-схемы. Но блок-схема алгоритма не является ментальной картой в том понимании, в котором мы пытаемся определить это понятие. Блок-схема не задействует чувственную зону памяти, интуицию, она обращается только к понятийной и абстрактной. А большой образовательный потенциал методики ментальных карт заложен именно в этих особенностях и преимуществах этой методики – использование для решения проблемы способностей обоих полушарий – и логического и интуитивного.

Опыт показывает, что при первоначальном изучении основных алгоритмических конструкций с учениками школы, а зачастую и со студентами младших курсов недостаточно изобразить блок-схему, соответствующую ей, необходимо, по-видимому, сделать ее более ментальной. И это, конечно же, уже будет не блок-схема в привычном понимании. Такую «оживленную» блок-схему, которая должна быть интуитивно понятна человеку, имеющему недостаточно высокий уровень алгоритмического мышления, мы предлагаем назвать ментальной алгоритмической картой [17].

Допустим, для изучения ветвления можно предложить следующую ментальную алгоритмическую карту – известный из сказок богатырь перед камнем и прорисовка двух ветвей со всевозможными последствиями и исходами, чтобы понять, как будут выглядеть исходные данные (полный сил богатырь, в одиночестве сидящий на коне) на выходе, после всей действий, которые совершатся по каждой из ветвей (богатырь с женой, богатырь пешком, без коня, ну, и мертвый богатырь); для изучения цикла – человечек, которому надо перенести с места на место 10 ящиков, а он может поднять только один – значит, ему надо 10 раз совершить одни и те действия. Рассмотрим подробнее ментальную алгоритмическую карту для изучения структуры «ветвление» и, соответственно, условного оператора. Ментальная

карта имеет 5 уровней абстракции. Первый житейский уровень – это «оживленная» (например, при помощи Flesh технологии) ситуация про сказочного богатыря.

На первом кадре – картинка, на который богатырь стоит перед камнем с указателями. Ему нужно сделать выбор, от которого зависит дальнейший исход: идти налево или направо. От выбора направления будет зависеть итог: либо богатырь будет богат, либо он будет женат. Третьего в нашей задачке не дано. Затем переходим от житейского представления задачи к более абстрактному. На этом уровне на ту же картинку наложены элементы блок-схемы. Богатырь является переменной, от значения которой будет зависеть направление движения – налево или направо. В центре камня появляется блок условия, которое при одном значении переменно ‘L’ будет истинным, при другом – ‘R’ – ложным. В зависимости от того, какое значение будет иметь переменная, и будет осуществлен выбор дальнейших действий. На третьем уровне абстракции элементы блок-схемы на той же картинке заменены операторами Паскаля. Это поможет ученику понять, что обозначает и что делает каждый фрагмент программного кода. На четвертом уровне картинка, которая была фоном ментальной схемы, уходит, и остается обычная блок-схема, которая соответствует алгоритму решения поставленной задачи. На заключительном пятом уровне на экране остается только программный код.

Такая «оживленная», в высшей степени ментальная, аппелирующая к чувственной, ассоциативной зоне памяти, алгоритмическая ментальная схема позволит осуществить плавный переход к высокому уровню абстракции в виде программного кода и будет способствовать более успешному формированию алгоритмического мышления, созданию ментальной схемы условного оператора, структуры «ветвления» у учеников [5].

Предполагаем, что использование подобных ментальных карт в процессе объяснения школьникам основных алгоритмических структур позволит повысить эффективность обучения программированию, сделает изложение учебного материала по этим темам более живым и наглядным, будет способствовать более успешному формированию у них алгоритмического мышления..

**Телесный подход к обучению программированию**

В настоящее время существует ряд российских и западных исследований, посвященных разработке методических подходов к обучению программированию, среди которых можно выделить: дифференцированный подход, системный подход, деятельностный подход,когнитивный подход, проблемный подход, семиотический подход [20].

Семиотический подход к обучению программированию заключается в целенаправленном развитии у обучающихся знаково-символических действий (замещения, кодирования, схематизации, моделирования). При этом учитывается, что обучающийся, уже знаком с некоторыми знаковыми системами (русским языком, дорожными знаками, позиционными системами счисления) и имеющиеся у него знания, опыт деятельности можно эффективно использовать при изучении особенностей языков программирования как знаковых систем [32].

Целесообразность применения семиотического подхода к обучению программированию может быть обоснована тем, что язык программирования изначально имеет знаковую природу. Как всякий искусственный язык он имеет «повторный перевод», другими словами, знания о мире формализуются, «переводятся» на обычный язык, а затем находят свое отражение в алгоритмических конструкциях, встроенных функциях и др., что влечет за собой постепенное развитие языка программирования[20].

Одним из принципов семиотического подхода является принцип учета ведущего канала восприятия. Согласно этому принципу, должны учитываться особенности учащихся с разными ведущими каналами восприятия (визуалами, аудиалами, дигиталами и кинестетиками). В этом отношении новым и мало изученным является телесный подход, в котором учитывается взаимосвязь телесных, кинестетических ощущений, восприятия и развития мышления.

Суть телесного подхода заключается в том, чтобы воздействовать преимущественно на кинестетические каналы восприятия, моторную область памяти, при том, что все остальные подходы к обучению воздействуют на дигитальные, аудиальные и/или визуальные каналы восприятия. Между тем, согласно телесному подходу, наше мышление «отелеснено», неразрывно связано с телесными ощущениями, и, следовательно, этот подход является единственно приемлемым для обучающихся с ведущим кинестетическим каналом восприятия и существенно более эффективным для всех остальных, поскольку задействует все каналы восприятия.

Телесный подход начал интенсивно развиваться в западной когнитивной науке примерно с начала 1990-х годов. Английское словосочетание «embodied cognition approach» точнее было бы переводить как подход с точки зрения «отелесненности» процесса познания, телесной облеченности всякого познающего существа. Такое уточнение имеется в виду, когда говорят неуклюже, но кратко: «телесный подход». Среди создателей нового подхода такие ученые, как Р. Бир, Р. Брукс, Т. Ванн Гелдер, Э. Кларк, Ж. Лакофф, П. Маес, Э. Прем, Э. Телен, Ф. Варела и ряд других [4].Другими словами, телесный подход фокусирует свое внимание на телесной организации познающего существа, так как то, что познается и как познается, зависит от телесности живого существа и его конкретных функциональных особенностей.

Возникновение и быстрое развитие телесного подхода было подстегнуто глубокой неудовлетворенностью ряда ученых доминировавшим с 60-х гг. вычислительным подходом (computationalapproach) к объяснению познавательных способностей человека и животных. Основной сферой приложения усилий представителей вычислительного подхода была проблема искусственного интеллекта. Идеалом виделась возможность построения системы, полностью имитирующей человеческий интеллект. В качестве модели для имитации брался компьютер. Предполагалось, что и мозг работает по принципам компьютера. Наглядным образцом такого рода устройства стали программы для игры в шахматы, основанные на просчитывании всех возможных ходов максимально далеко вперед.

Создателей этих программ радовало и обнадеживало то, что возможности компьютера в чем-то даже превосходят возможности человеческого интеллекта. Казалось, стоит еще и еще повысить тактовую частоту процессора, добавить гигабайтов памяти, объединить множество компьютеров в единую сеть, запустив по возможности параллельно, что уподобило бы их нейронной сети мозга, одновременно выполняющей множество задач – и цель достигнута.

Сторонники телесного подхода выдвинули следующее возражение резервы на этом пути, конечно, есть, но сам путь обходит стороной главное.

Их возражения в адрес вычислительного подхода в обобщенной форме можно сформулировать следующим образом:

1. Вычислительный подход сводит функции познания к функциям чистого, абстрактного интеллекта. Интеллект в этом случае существует как бы вне тела, вне физического организма, взятого в его естественном функционировании и движении и в окружении других материальных тел. Фактор телесной облеченности субъекта восприятия и мышления в вычислительной модели не только излишен, но и вреден, поскольку только затемняет и искажает деятельность чистого интеллекта. Тем самым эта

модель лишается связи с реальностью, а потому объяснительной и эвристической силы.

2. Вычислительный подход рассматривает когнитивные функции, причем сведенные лишь к интеллектуальным функциям, в их наличной данности, в полностью развитом виде, игнорируя и общее эволюционное происхождение этих функций (процесс филогенеза), и постепенность их формирования в процессе индивидуального развития особи (процесс онтогенеза).

3. Мыслительные операции человека, в представлении вычислительного подхода, строятся по принципу символического представления (репрезентации – поэтому вычислительный подход нередко называют также репрезентатистским), который лежит в основе работы компьютера, где входные данные переводятся на особый символический язык, посредством которого обрабатываются. Это означает, что если процесс «вне» головы, вызвавший когнитивный акт, можно объяснить как физический динамический процесс, то процесс «в» голове следует объяснять по законам семантики, т.е. смыслового отношения одной системы символов с другой системой символов. Тем самым процесс познания, а с ним и мир в целом, оказывается удвоен, разорван, по меньшей мере, на две несводимые реальности – физическую и семантическую.

Черты самоорганизации и самодвижения в компьютерах весьма ограничены. Моделируя когнитивные функции по образцу функций компьютера, мы лишаем этих черт и естественное функционирование мозга, а потому нуждаемся во введении в модель мозга внешней движущей силы или программы, посредством которой осуществляются самотестирование системы и ее самоконтроль.

Ведь, во-первых, если подходить к пониманию деятельности мозга по семантической схеме, вряд ли можно утверждать, что семантические системы, в отличие от физических динамических систем, способны к

спонтанному движению, в том числе «переведению» себя в другую систему символов. Во-вторых, с технической точки зрения, компьютерные операции осуществляются в тактовом режиме, а не непрерывно: с каждым новым тактом, отмеряемым процессором, изменяется распределение электромагнитных состояний, что в более широком временном масштабе выглядит как непрерывная работа компьютера. Но такого рода движение есть не подлинное динамическое движение, а последовательность сменяющих друг друга статических состояний. Тем более не есть оно самодвижение, поскольку осуществляется только путем «подталкивания» со стороны процессора, и одно статическое состояние никоим образом не способно самопроизвольно перетекать в другое состояние, разве что как сбой системы.

В противовес вычислительному подходу была выдвинута теоретическая концепция, базирующаяся на следующих утверждениях.

1. Познание телесно, или «отелесненно». То, что познается и как познается, зависит от строения тела и его конкретных функциональных особенностей, способностей восприятия и движения в пространстве. Устроены тела по-разному – потому и познают тела по-разному. Если раньше гносеологи говорили, что познание теоретически нагружено (т.е. то, что мы видим, во многом определяется имеющимися у нас теоретическими представлениями), то теперь можно сказать к тому же, что познание телесно нагружено.

Нельзя понять работу человеческого ума, если брать ум абстрагированнным от организма, телесности, определенным образом обусловленного восприятия посредством конкретных органов чувств.

2. Познание ситуационно. Познающее тело погружено в более широкое – внешнее природное и, в случае человека, социокультурное окружение, оказывающее свои влияния.

3. Познание инактивировано (enactedcognition). Познание осуществляется в действии и через действия. Через действия формируются

когнитивные способности, как видовые, так и индивидуальные. Когнитивная активность в мире создает и саму окружающую по отношению к познающему существу среду – в смысле отбора, вырезания им из мира именно и только того, что соответствует его телесным потребностям, когнитивным способностям и установкам.

4. Познавательные системы есть динамические и самоорганизующиеся системы. В этом функционирование познавательных систем принципиально сходно, единосущно функционированию познаваемых природных систем, т.е. объектов окружающего мира. Потому и мир остается един, а не разорван на динамическую «внеголовную» и семантическую «внутриголовную» половины.

С точки зрения сторонников вычислительного подхода, мир поделен на две несводимые реальности: символическую, представленную внутри головы, и физическую – вовне. Поэтому для них встает проблема: как соединить обе эти реальности или объяснить одну через другую.

Телесный подход говорит о единстве реальности и принадлежности к ней как агента со свойственными ему когнитивными процессами, так и внешней ему среды. Эта единая реальность имеет физический характер, а процессы в ней являются динамическими процессами самоорганизации.

В телесном подходе присутствуют два противоположных мотива в трактовке роли тела в процессе познания. С одной стороны, только через тело и его движение формируется и осуществляется познание. Но, с другой стороны, тело привносит такие искажающие влияния, от которых приходится избавляться. Идея чистого интеллекта присутствует и как предмет критики, с точки зрения ее нежизненности и научной несостоятельности, и как идеал. Сам по себе недостижимый, именно он задает направленность вектора объективации. Такое противоречие можно снять в уме с помощью гегелевской диалектической триады: до помещения в тело чистый интеллект есть выхолощенное представление, но после помещения в тело и с учетом

такой помещенности – на новом витке спирали, он становится важным притягивающим аттрактором и задающим критерием.

В телесном подходе получила развитие тема категоризации человеком мира исходя из физического существования его тела. Сетка этих категорий как бы накладывается на мир, и только согласно ее клеточкам мир и воспринимается. Категории эти не случайны и не произвольны, как полагалось ранее. Они происходят из телесного опыта, а он имеет повторяющийся и устойчивый характер [11].

М.А. Степанов отмечает, что телесный подход открывает новую перспективу на культурный мир и человека в нём, где главное отличие проходит через смену отношения к месту и времени: истолкование традиционно происходит после события, телесное познание требует готовности к будущему и решительности в спонтанном акте его реализации. Или, иначе говоря, компетентности и способности к действию [34].

Современная школа ориентирована преимущественно на логическую и аудиальную стороны мышления. По мнению профессора психологии Гарвардского лингвистическим университета ГовардаГарднера[24], учебные заведения созданы для детей с логико-математическими лингвистическим типом мышления. Однако в современных психологических исследованиях кроме них выделяют телесно-кинестетический, визуально-пространственный, внутриличностый типы мышления.

Согласно полученным психологами данным, лингвистический и логический (математический) типы интеллекта вовсе не преобладают среди молодежи, тем более среди подростков или младших школьников. Получается, что большинство детей разного возраста остается за бортом качественного образования.

Практикующие психологи дают свои рекомендации по особенностям воспитания и обучения детей с разными типами интеллекта[24]. Так, у частности, если у ребенка преобладает телесно-кинестетический интеллект,

то обработка информации у него происходит на уровне тактильных сигналов. У такого ученика умелые руки и отличное чувство баланса. Он лучше всего чувствует себя в случае, когда надо много времени посвящать физической деятельности, ремеслу, он легко управляет предметами. Для максимально эффективного обучения ему необходимо прикасаться, двигаться и обрабатывать поступившую информацию посредством тактильных ощущений.

Если ребенок кинестетик, то он предпочитает учиться, выполняя задание. Самым эффективным способом обучения для таких детей, как правило, является экспериментирование, а также разработка способов решения поставленной задачи. Ребенок-кинестетик лучше всего запоминает и усваивает информацию посредством прикладного опыта, в движении, в ролевых играх, в игре и групповых мастерских, в импровизации, в работе с материалами.

Представляется, что если строить систему обучения, опираясь на эти советы, использовать средства обучения, нацеленные на активизацию кроме визуальной, аудиальной и абстрактной еще и моторной зоны памяти, процесс обучения программированию будет проходить более эффективно.

## 1.2 О*собенности изучения ООП*

Современные образовательные стандарты предъявляют высокие требования к предметной подготовке учителя. Учитель информатики должен владеть всеми современными технологиями программирования, поэтому в курс «Языки и методы программирования» в педагогическом вузе включено изучение не только структурного программирования, которое изучается в школе, но и логического, функционального, объектно-ориентированного, параллельного и др. современных технологий программирования.

**Возникновение и основные принципы ООП**

В 1967 г, практически сразу после появления языков третьего поколения при разработке программ, реализующих модели сложных систем разработчики столкнулись с задачей, решение которой без декомпозиции оказалось невозможно. Это обстоятельство натолкнуло на идею несколько преобразовать постулат фон Неймана о том что данные и программы неразличимы в памяти машины. Они решили: пусть данные и программы если не станут одним и тем же, то сильно к этому приблизятся.

Попытки обосновать декомпозицию первоначально отличались чрезмерной размытостью подхода, его объяснение скорее на уровне понимания и интуиции, чем на уровне правил . Усилия многих программистов и системных аналитиков, направленные на формализацию подхода, увенчались успехом.

Были разработаны три основополагающих принципа того, что потом стало называться объектно-ориентированным программированием (ООП):**наследование, инкапсуляция, полиморфизм**.

Результатом их первого применения стал язык Симула-1 (Simula-1), в котором был введен новый тип - *объект*. В описании этого типа одновременно указывались данные (*поля*) и процедуры, их обрабатывающие

- *методы*. Родственные объекты объединялись в *классы*, описания которых оформлялись в виде блоков программы. При этом класс можно использовать в качестве префикса к другим классам, которые становятся в этом случае подклассами первого. Впоследствии Симула-1 был обобщен и появился первый универсальный ООП ориентированный ЯП - Симула-67 (67 - по году создания).

Пределом объектной ориентации принято считать Смолток (SmallTalk), в котором доступ к полям объектов возможен только через их методы.

Как выяснилось, ООП оказалось пригодным не только для моделирования (Simula) и графических приложений (SmallTalk), но и для большинства других приложений, а его приближенность к человеческому мышлению и возможность многократного использования кода сделали его одной из наиболее бурно используемых концепций в программировании.

Разберем три принципа, которые стали почти достаточными для реализации концепции ООП. Предварительно введем определения слов "объект" и "класса".

**Объект** совокупность (разнотипных) данных (полей объекта), **физически** находящихся в памяти ЦВМ, и алгоритмов, имеющих доступ к ним.
Каждый объект может обладать *именем* (идентификатором), используемым для доступа ко всей совокупности полей, его составляющих. В предельных случаях объект может не содержать полей или методов.

**Класс** - тип (описание структуры данных и операций над ними), предназначенный для описания множества объектов.
Каждый класс может иметь *подклассы* - классы, обладающие всеми или частью его свойств, а так же собственными свойствами. Класс, не имеющий ни одного представителя (объекта) обычно называют *абстрактным*.

**Инкапсуляция**

Несмотря на непривычность слова просто связывание полей и методов в одну структуру (складывание их в одну "капсулу"). Это удобно, хотя и без остальных двух принципов никакого нового *качества* программирования не дает. Действительно, если объединить данные хотя бы с алгоритмами доступа к ним, то программист окажется независимым от представления данных в объекте: объект становится абстракцией представления своих собственных данных.

В более общем случае объекту можно приписать свойства (методы), абстрагирующие не только представление, но и придающие объекту другие свойства, к примеру способность отображаться.

Теоретически принцип инкапсуляции применим как к отдельным объектам, так и к классам. В случае классов с методами объединяются не сами данные, а *структуры данных*, и объединение с конкретными данными происходит в момент создания объектов данного класса. На практике же большинство ОО языков просто не позволяют создавать объекты, не создав предварительно класса.

**Наследование**

Этот принцип относится только к классам объектов. Наследование означает, что каждый объект может иметь наследников, каждый из которых будет обладать всеми полями и методами своего предка. Кроме того, как правило, классы-наследники совместимы по типу со своими предками (к сожалению это справедливо не для всех ОО языков).

Наследование бывает двух видов:

- *одиночное* - когда каждый класс имеет одного и только одного предка;
- *множественное* - когда каждый класс может иметь любое количество предков.

Множественное наследование обладает более мощными возможностями: в одном классе-наследнике объединяются свойства (поля и

методы) множества различных классов. К примеру один из предков может рисовать себя, а другой - производить вычисления. представитель их наследника смогут делать и то, и другое.

**Полиморфизм**

Этот принцип неразрывно связан с наследованием и гласит, что каждый класс наследник может обладать не только свойствами, унаследованными от предка, но и своими собственными. В частности, свойства предка могут быть *перекрыты* наследником - на место свойств предка могут быть подставлены свойства наследника.

Существование принципа полиморфизма является естественным следствием существования принципа наследования: наследование без изменения набора свойств не имеет смысла. Кроме того, без полиморфизма невозможно реализовать объединение различных объектов (классов) по некоторому набору свойств (невозможно абстрагироваться от части свойств объектов), а без этого теряется весь смысл подхода.

Для наглядного представления структуры классов и их взаимоотношений в логике создаваемого программного продукта существует возможность их описания в виде UML-диаграмм.

**UML -диаграммы**

**Unified Modeling Language (UML)- унифицированный язык моделирования**, применяется на различных этапах разработки программного обеспечения. Был создан в середине 90-х годов (1994-1996). На данный момент есть спецификация uml 1, uml 2. Именно версию 2 мы будем рассматривать ниже.

Для прочтения UML-диаграмм нужно уметь читать или знать условные обозначения UML.

В UML программе представляются диаграммами. В UML диаграммах представляется общая архитектура программы или её некие особенности. От сюда можно сделать ввывод, что в UML-диаграммах создается модель

программы, которую нужно сделать. Язык UML является высоким уровнем абстракции, поэтому программы, сделанные в UML, впоследствии можно реализовать на различных языках,  в котором присутствует достаточно возможностей объектно-ориентированного программирования.

UML  может реализовываться на различных этапах разработки ПО: 1. во время проектирования ПО, 2. во время кодирования на конкретном языке. UML используется не только программистами, но и, например как, менеджерами в компаниях, разрабатывающих ПО.

Можно на примере убедиться, взять три класса. Каждый класс состоит из 300 строк кода. Между классами определены сложные связи. Уследить за кодом будет сложно. А например, если классы представить в UML-диаграммой, то все классы и связи можно будет проследить на одной большой диаграмме.

На первый взгляд спецификация Unified Modeling Language кажется очень сложной. Но на самом деле читать UML- диаграммы легко, главное разобраться с условными обозначениями.

**Диаграммы классов UML (Class diagram)**

Для удобства мы будем пользоваться диаграммами классов (Class diagram), т. к. в данном типе диаграмм показывается взаимодействие программ.

**Элементы диаграмм UML**

Диаграммы UML состоят из элементов. Элементы представляются прямоугольниками, в которых записывается имя элемента. Например, изобразим в UML-диаграмме какой-нибудь класс(рис. 4):

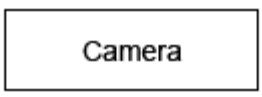



Рис.4 Класс Camera.

**Комментарии в UML**

Комментариев в UML обозначается прямоугольник с "загнутым" правым верхним уголком. Пунктирной линией обозначается к какому элементу принадлежит комментарий(Рис.5):

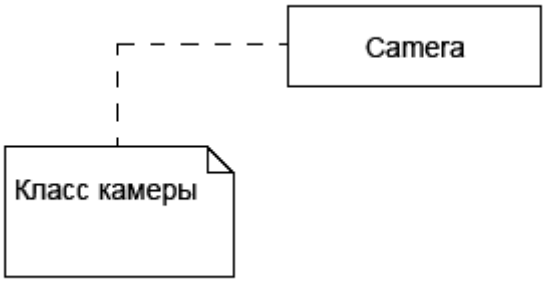



Рис.5 Комментарий к классу Camera.

**Атрибуты (attribute) и операции (operation) в UML-диаграммах**

В C++ переменная, принадлежащая классу, называется полем класса(переменной-членом), то в UML такая переменная называется атрибутом. Также и с функцией/методом класса - в UML это операция.

Для атрибутов и операций в элементах отводится отдельный блок. Каждый блок делится горизонтальной чертой. Например, для класса Camera элемент с атрибутами и операциями представлена на рисунке 6:

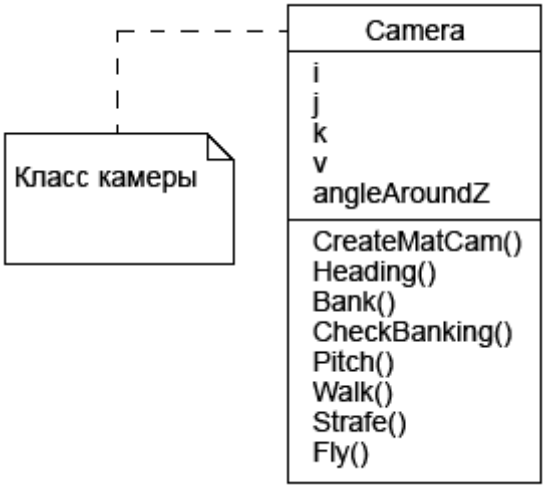



Рис.6 Обозначение элемента с атрибутами и операциями.

Через двоеточие задается атрибут(как и аргумент операции) (Рис. 7)

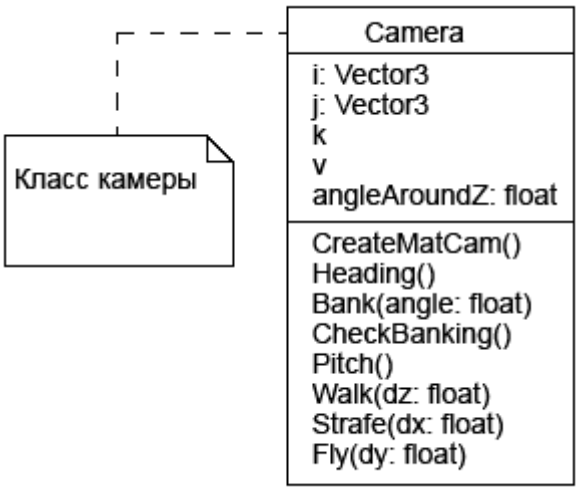



Рис. 7 Тип атрибута

Здесь представлены все достоинства UML. В Unified Modeling Language необязательно расписывать **все** детали классов. Это будет сделано при написании кода на языке C++. В UML-диаграмме можно не прописывать незначимые детали. Например, в диаграмму элемента можно добавить только те операции или атрибуты, которые для данной диаграммы важны.

**Видимость атрибутов и операций в UML: +, -, # (спецификаторы доступа)**

Спецификатора доступа языка C++ (public, private, protected) в UML отображаются символами + (public), - (private), # (protected), которые записываются перед именем атрибута или операции. Есть еще вариант с ключевыми словами public, private, protected. На рисунке 8 можно сравнить два варианта записи:

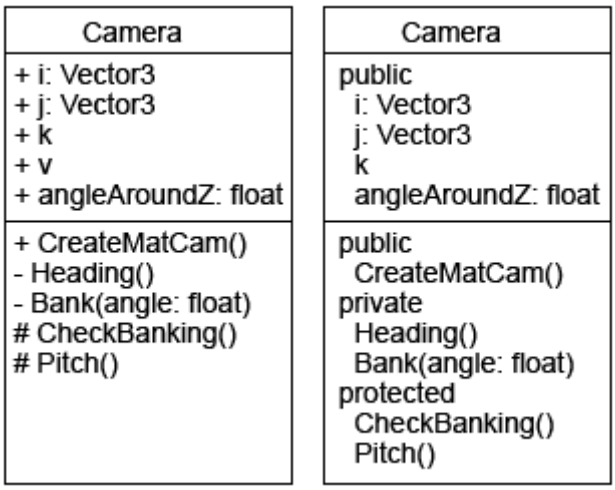



Рис.8 Два варианта записи спецификатора.

Значение спецификаторов доступа является: public – поля или методы класса видны снаружи класса. Т.е. к ним получают доступ объекта класса. private – поля или методы класса видны только внутри определения класса. protected – поля или методы класса видны в определении самого класса и в определениях производных классов.

**Отношения между классами в ООП (UML, C++)**

В программах между классами присутствуют различные виды взаимодействия (или связи): один класс может быть производным другого, третий может содержать объект четвёртого в виде поля. Для различных видов взаимодействия в UML есть специальные обозначения.

**Ассоциация(объединение/связь) (association)**

Первый вид связи - association. С русского переводится по-разному: ассоциация, связь, объединение. Наилучший вариант перевода, является связь, но я это слово предназначено для всех видов взаимодействия классов. Поэтому для обозначения association лучше всего пользоваться словом ассоциация.

Ассоциация – является очень слабым видом связи. Возникновение ассоциации вызывается тогда, когда один класс вызывает метод другого или если при вызове метода- аргумента передаётся объект другого класса.

На UML-диаграммах ассоциация отображается сплошной линией.

Ниже приведен простой класс:

```
class MonstAr
{
private:
attack(int damage) // damage - урон
{}
};
```

Стандартные типы C++ и есть класс. На диаграмме UML представлено взаимодействие классов MonstAr и int (Рис.9):

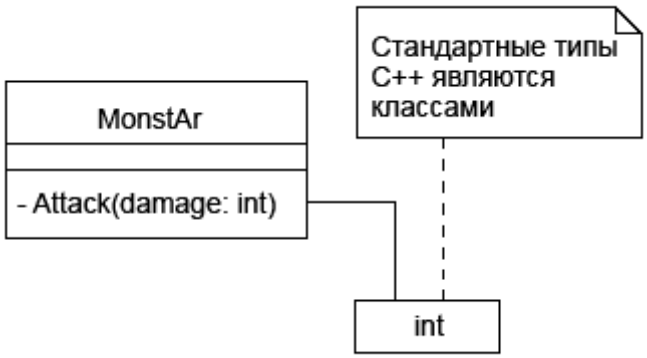



Рис.9 Взаимодействие классов MonstAr и int.

На этой диаграмме показано отсутствие атрибутов у элемента.

Иногда при ассоциации, если присутствует значение, показывают направленность . В спецификации UML используется слово navigable. Но лучше всего, на русском языке нужно *направленность* , так как это слово отражает правильную суть. Направленность обозначается с помощью стрелочки(Рис.10). Так же нужно не забывать оп равильности нанесения стрелочки, это играет важную роль на нанесения диаграммы.

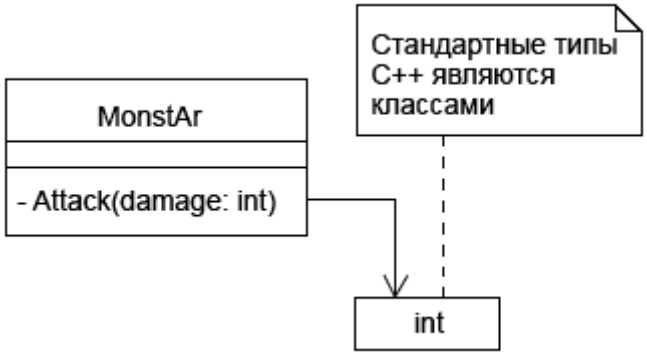



Рис.10 Направленность

В данной диаграмме стрелочка направлена на int. В данном случае направленность ассоциации показывает нам, что в методе MonstAr::Attack используется объект типа int.

**Обобщение (generalization)**

Для обозначения наследования в UML используется обобщение-generalization, этот термин взят из UML- диаграммы. Например:

```
MonstAr
{
private:
attack(int damage) // damage - урон
{}
};
BigMonstAr : public MonstAr // большой (big) MonstAr
{
// определениекласса
};
SmallMonstAr : public MonstAr // маленький (small) MonstAr
{
// определение класса
};
```

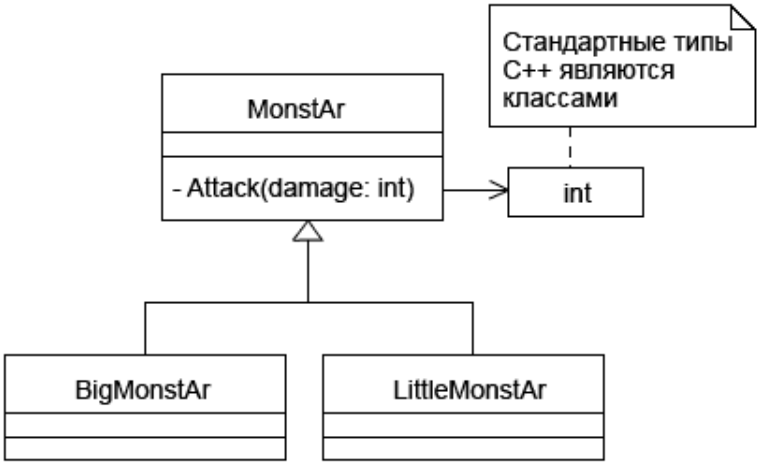



Рис.11 UML- диаграмма с обобщением.

При обобщении обозначается сплошной линией, как представлена на рисунке 11. Стрелочка рисуется с пустым треугольником на конце.

В UML используется слова *обобщение* (generalization), а не *наследование* , так как в данном виде связи один из классов (базовый) является общим, а остальные классы (производные) - более специализированными.

**Aggregation - агрегация, агрегирование, включение в UML**

Следующий тип связи между классами – aggregation(с латинского на присоединение. По-русски это будет агрегация, агрегирование или соединение частей. Лучше всего использовать слово *агрегация* .

Итак, в UML агрегация отражает, когда объект одного класса(атрибутом другого). Например:

```
class MonstAr
{
public:
int a;
};
```

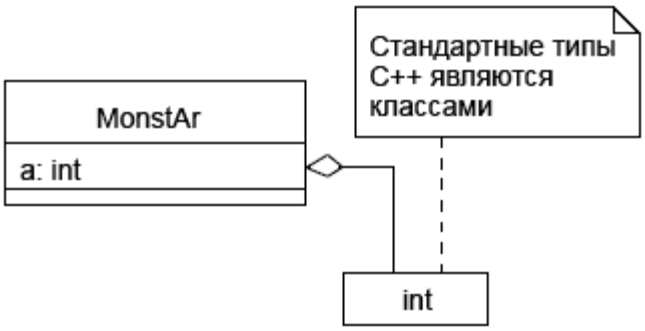



Рис 12. На диаграмме представлена агрегация.

**Композиция классов - composition в UML**

Композиция классов – является сильной связью между классами в отличии , чем агрегация. Между композицией и агрегацией довольно тонкая грань. Особенностью композиции является то, что объекты, из которых создаётся композиция, могут принадлежать только классу, с которым они образуют композицию. При этом время жизни объекта и класса, в который встраивается объект, совпадает.

Можно взять два примера  из жизни. Возьмем dvd-привод и диски, которые он читает, образуют агрегацию. Диски можно свободно менять. Следующим примером композиции может служить мука и хлеб. Извлечь муку никак нельзя из хлеба. На этих двух примерах хорошо представлена разница между агрегацией и композицией: компоненты собранные агрегацией можно разъединить, а с композицией этого сделать невозможно.

Одним из признаков агрегации является использование указателей. И наоборот, если при связи классов указатели не используются, то существует большая вероятность, что перед нами композиция классов.

```
class Claws; // claws - когти
class MonstAr
{
public:
Claws MonstArClaws;
};
```

В данном случае у монстра "есть когти" (взятые в отдельном классе). Здесь хорошо видна композиция классов: нельзя от монстра разделить с его когтями (он будет сильно недоволен). В UML композиция выглядит вот так:

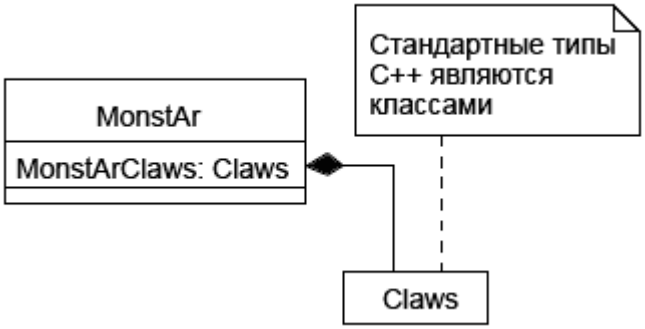



Рис.13 UML композиция.

На диаграммах композиция показывается закрашенным ромбом(Рис.13).

**Реализация - realization в UML**

Последнее отношение, которое мы рассмотрим, будет realization - реализация. Данная связь показывает отношение: класс - объект.

На диаграмме реализация показывается пунктирной линией и незакрашенной стрелочкой(Рис.14):

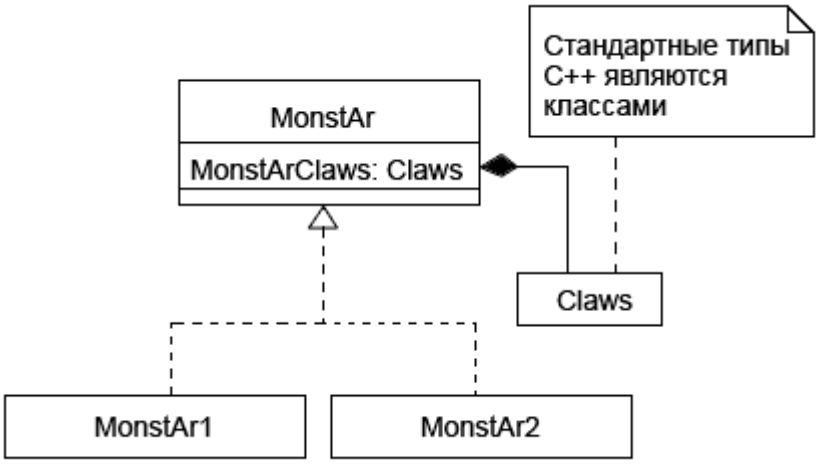



Рис.14 Диаграмма реализация

**Пример диаграммы классов.**

UML диаграмма классов показывает список классов в системе (или подсистемы) и отношения между классами. Проект моделирования

показывает также атрибуты и методы классов.На рисунке 16представлена представлена простая диаграмма классов, которая визуализирует. Частичное обобщение (Party generalization) и показывает, как к этому обобщению можно легко присоединять класс компаний и класс людей. Такая диаграмма представлена на рисунке 16.

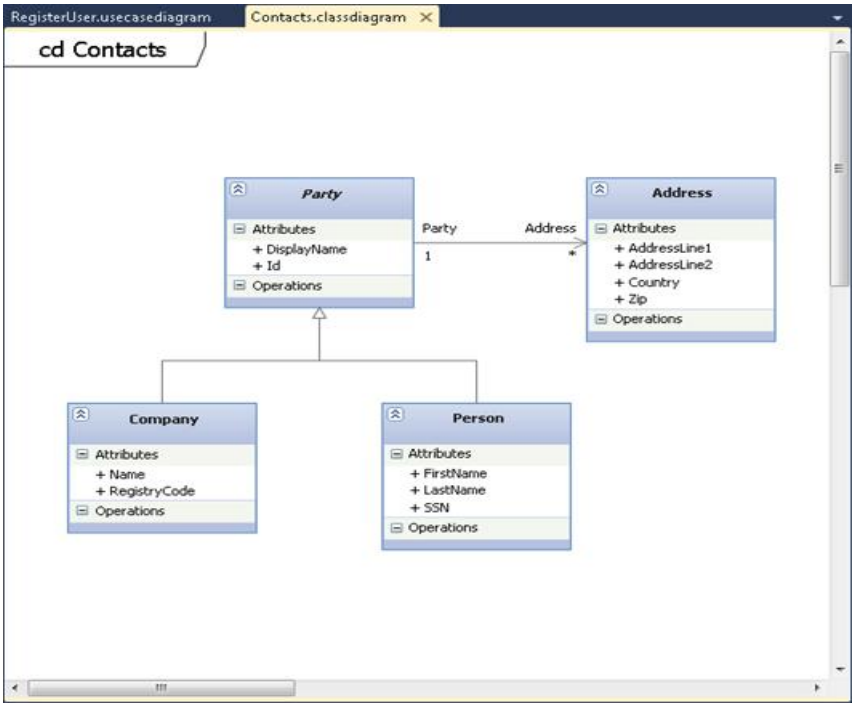



Рис.15 Пример диаграммы классов.

Однако, несмотря на то, что UML-диаграммы являются специально созданным средством для наглядного представления структуры класса и взаимосвязи между классами в ООП, они имеют высокий уровень абстракции, и, кроме того, являются статичным. С их помощью невозможно отобразить алгоритмы работы с классами.

**Проблемы, возникающие при обучении ООП**

В методологии объектно- ориентированного программирования (ООП) за пятьдесят лет существования сформированы знания и накоплен опыт по качественной разработке объектно-ориентированного программного

обеспечения. В настоящее время методология ООП становится одной из центральных в подготовке будущих специалистов, существует потребность в ее глубоком изучении, но обучение ООП в вузах началось сравнительно недавно. Студенты испытывают различные трудности в процессе обучения ООП, связанные с:

формированием представлений об объектно-ориентированном программировании;

осознанием основных принципов объектно-ориентированного программирования – инкапсуляции, наследственности, полиморфизма;

изучением объектной декомпозиции на практике и др

И особенно сложно перестроить стиль мышления с процедурного на объектный.

С одной стороны, основной причиной проблем в обучении является недостаточная теоретическая разработанность методики обучения ООП. С другой стороны, для студентов трудности изучения ООП связаны со сложностью процесса разработки программного обеспечения.

Обучению объектно-ориентированному программированию и проектированию посвящены работы зарубежных исследователей, таких как Г. Буч «Объектно-ориентированный анализ и проектирование», Б. Мэйер «Объектно-ориентированное конструирование программных систем». Книга «Унифицированный процесс разработки программного обеспечения» авторами которой являются: А. Якобсон, Г. Буч, Дж. Рамбо, И. Грэхем, М. Фаулер, а также исследования отечественных ученых Г.С. Ивановой, Т.Н. Ничушкиной, Е.К. Пугачевой, И.А. Бабушкиной и др.

Методология ООП многогранна и нестандартна. Отсутствие единой, общепринятой точки зрения на объектно-ориентированный подход является как ее преимуществом, так и недостатком. Преимущество в том, что в процессе обучения можно рассматривать различные интерпретации ООП. Недостаток в том, что отсутствие единой точки зрения на ООП может

привести к ее формальному и ограниченному изучению. Рассмотрение принципов ООП без обучения объектной декомпозиции, объектно-ориентированному проектированию и реализации на практике преимуществ ООП, не способствует формированию у студентов необходимых представлений об ООП.

Студенты испытывают большую часть на начальном этапе обучения: в процессе формирования представлений об основах ООП. Оттого, как формируются эти представления, будет зависеть все дальнейшее изучение методологии ООП.

Петров А. Н. в своей статье под названием «Особенности методики обучения студентов объектно-ориентированному программированию и проектированию» [32] говорит, что на начальном этапе обучения студентов ООП и объектно - ориентированному проектированию использовать презентации. Анимация, выделение цветом строк программного кода и соответствующих частей элементов диаграммы классов языка UML помогут студентам лучше понять взаимосвязи объектно-ориентированного программного кода и диаграммы классов языка UML. Использование презентаций на начальном этапе обучения ООП позволит студентам начать применять язык UML в объектно-ориентированном проектировании и создавать на основе него объектно - ориентированный программный код.

У большинства студентов сформирован алгоритмический императивный, процедурный стиль мышления. Смена стиля мышления у студентов обычно происходит тогда, когда они начинают понимать преимущества, которые предоставляет методология ООП. Объектно-ориентированный подход позволяет: решать проблемы построения сложных систем; улучшать сопровождение программного обеспечения; расширять и масштабировать программный код; создавать повторно используемый программный код. Эти преимущества являются мотивирующим фактором для изучения студентами методологии ООП. Освоение объектной

декомпозиции является одним из решающих факторов, который может привести к изменению стиля мышления студентов со структурного на объектный.

На начальном этапе обучения ООП не должна преобладать алгоритмическая декомпозиция по отношению к объектной декомпозиции. Рассмотрение объектной декомпозиции в начале обучения ООП желательно сократить до минимума и продолжить ее изучение тогда, когда у студентов будет сформировано представление об основах ООП, и они смогут реализовывать объектную декомпозицию на практике.

Осуществление объектной декомпозиции позволит студентам абстрагироваться от программного кода в целом и сконцентрироваться на определенных классах, некоторых отношениях между классами. «В основе любого подхода к программированию лежит понятие декомпозиции (разбиения на части) сложных систем с целью последующей реализации в виде отдельных небольших (до 40 - 50 операторов) подпрограмм». [29]

При осуществлении объектной декомпозиции рекомендуется придерживаться следующих правил по определению классов:

1. объект должен быть простым и понятным с точки зрения его структуры;
2. объект не должен включать в себя несколько абстракций, поэтому имеет смысл разделить этот объект на несколько объектов;
3. объект должен быть "самодостаточным".

Объектная декомпозиция осуществляется до тех пор, пока не будут определены объекты, имеющие четкую структуру данных, поведение, соответствие их решаемой задаче и установленным отношениям между другими объектами. Студенты в процессе изучения объектной декомпозиции учатся самостоятельно осуществлять поиск и "отбраковку" классов, обосновывать выбор классов и отношений между ними. Определение

объекта начинается с выяснения того, что это за объект и какую роль он играет в данном случае. Роль объекта определяет его атрибуты и операции.

В начале объектно-ориентированного проектирования не следует фиксировать роль какого-либо объекта, так как это может ограничить выбор ролей для объектов, связанных с данным объектом. Поэтому желательно рассмотреть наибольшее количество возможных ролей, которые могут быть применены к объектам. Роль объекта устанавливается в зависимости от выбираемых ролей для объектов, связанных с ним. Таким образом, роли объектов имеют большое значение для реализации объектной декомпозиции.

Студенты должны учиться: применять свои знания в реальных ситуациях; выражать свои идеи на языке UML; расширять сферу возможного применения ООП. Для этого рекомендуется решать сюжетные задачи и задачи, имеющие объекты, прототипами которых являются реально существующие объекты.

Итак, в ходе обучения объектно-ориентированному программированию и проектированию необходимо опираться на методические рекомендации ведущих специалистов в области методики обучения ООП, таких как И.Н. Аржанов, Е.В. Баранова, А.Г. Кирилов, Н.А. Мещерякова, М.С. Орлова, А.Н. Петров, Ю.А. Петрова, А.Г. Степанов по формированию представлений об ООП, изучению объектной декомпозиции и преимуществ ООП на практике. Студенты должны учиться обосновывать принимаемые решения и находить подходящий вариант построения диаграммы классов UML; решать сюжетные задачи и задачи, имеющие объекты, прототипами которых являются реально существующие объекты. Тогда процесс обучения ООП будет соответствовать современным требованиям.

Но анализ экспертных заключений преподавателей о трудностях, возникающих при обучении ООП и мой собственный опыт изучения ООП показывает, что традиционно используемых блок-схем как средств

формального представления алгоритмов решения задач и UML-диаграмм недостаточно для понимания, особенно при изучении такой сложной технологии как ООП, нужно еще одно звено – между алгоритмическим мышлением на житейском, повседневном уровне и алгоритмическим мышлением на формализованном уровне, необходимом для успешного изучения ООП. [5]. Таким недостающим звеном и принципиально новым средством обучения программированию могут служить алгоритмические ментальные карты.

### *1.3. Уточнение понятия объектного стиля мышления.*

На современном этапе развития информатики для успешного взаимодействия с компьютером необходим стиль мышления, который можно назвать **объектным. [37]** Объектный стиль мышления предполагает умение разделить сложную систему на объекты и выстроить их иерархию, т.е. произвести объектную декомпозицию системы, а затем описать поведение этих объектов. Основной операцией при таком стиле является структурная модель, разложение объектов. Всевозможные классификации по различным логическим основаниям и логические методы формирования понятий составляют значительную часть методов, используемых при таком стиле мышления. При описании событий используется алгоритмическая декомпозиция системы и необходим алгоритмический стиль мышления.

Компонентами объектного стиля мышления являются:

1. Анализ предметной области задачи и выделение объектов (реальных и абстрактных), построение их иерархии.
2. Выделение основных событий.
3. Реализация процессов обработки событий.
4. Анализ поведения системы и коррекция объектной модели и алгоритмов обработки событий в случае несовпадения полученного результата с предполагаемым [13]..

К специфическим свойствам объектного стиля мышления относятся:

- высокий уровень абстрактности, который заключается в выделении существенных характеристик объекта и абстрагировании от его свойств, несущественных для решения конкретной задачи;
- осознанная закрепленность в языковых формах, предполагающая отражение построенной объектной модели задачи на некотором формализованном языке;

- целостность восприятия сложной системы, представление ее в виде совокупности взаимодействующих объектов.

Объектный стиль мышления предполагает способность студента к умению в реально существующем объекте определить свойства этого объекта (данные) и выделить методы (алгоритм их обработки). Определить класс объекта.

Структурная модель объектного стиля мышления представлена на рис.16

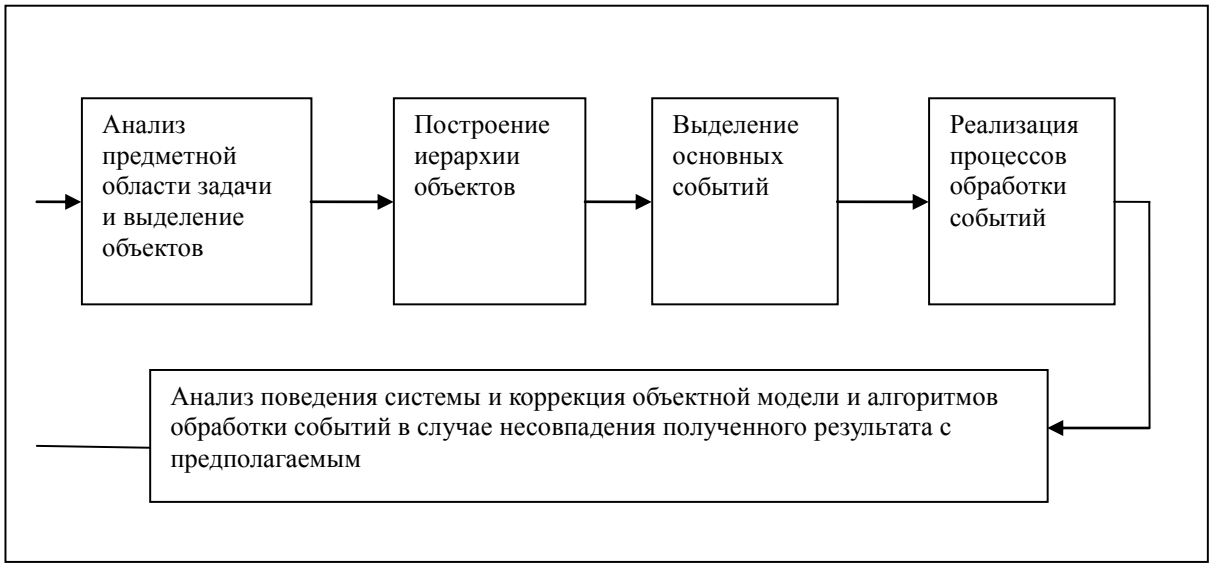



Рисунок 16. Структурная модель объектного стиля мышления

Уточнив таким образом понятие объектного стиля мышления на основе ментального подхода, мы можем целенаправленно формировать его в процессе обучения ООП, без чего это обучение не будет являться эффективным.

***Выводы по 1 главе***

– Описана сущность ментально-телесного подхода к обучению.

– Описано возникновение методологии ООП, его основные принципы, UML-диаграммы как средство визуализации в ООП

– Выделены особенности обучения объектно-ориентированному программирования

– Выявлено, что на успешность обучения объектно- ориентированному программированию влияет уровень сформированности объектного стиля мышления.

– Уточнено понятие объектного мышления, построена его структурная модель.

# ГЛАВА 2. УСЛОВИЯ РЕАЛИЗАЦИИ МЕНТАЛЬНО-ТЕЛЕСНОГО ПОДХОДА К ОБУЧЕНИЮ ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ

## *2.1 Диагностика уровня сформированности объектного стиля мышления.*

Определенный уровень сформированности объектного мышления является необходимым условием успешного обучения объектно-ориентированному программированию. Следовательно, для разработки эффективных методик обучения ООП необходимо иметь соответствующие диагностики уровня его сформированности.

Особенность данной диагностики на наш взгляд должна заключаться в том, что мы должны диагностировать не уровень усвоения знаний по ООП, а именно особенности мышления. С этим же связана сложность ее создания.

Вопросам уточнения понятия «объектное мышление» посвящены работы Газейкиной А.П. [Газейкина, 2006], Нигматулиной Э.А. [Нигматулина, 2011, стр.83], Степановой Т.А. [Степанова, 2015, стр.248].

Согласно этим исследованиям, компонентами объектного стиля мышления являются:

1. Анализ предметной области задачи и выделение объектов (реальных и абстрактных), построение их иерархии.
2. Выделение основных событий.
3. Реализация процессов обработки событий.
4. Анализ поведения системы и коррекция объектной модели и алгоритмов обработки событий в случае несовпадения полученного результата с предполагаемым. [Газейкина, 2006].

К специфическим свойствам объектного стиля мышления относятся:

1. Высокий уровень абстрактности, который заключается в выделении существенных характеристик объекта и абстрагировании от его свойств, несущественных для решения конкретной задачи

2. Осознанная закрепленность в языковых формах, предполагающая отражение построенной объектной модели задачи на некотором формализованном языке.

3. Целостность восприятия сложной системы, представление ее в виде совокупности взаимодействующих объектов. [4]

Следовательно, задания на диагностику уровня сформированности объектного мышления должны быть направлены на проверку способности выделить объект, увидеть его место в иерархии (какие объекты будут являтся его родителями, а какие – потомками, определить свойства и методы выделенного объекта.

В соответствии с этим, предлагаются следующие уровни сформированности объектного мышления:

1. Начальный уровень: способность выделить объект.

2. Средний уровень: умение описать свойства и методы, присущие донному объекту.

3. Высокий уровень: умение увидеть объекта в иерархии, определить, какие свойства и методы будут наследоваться им от родителя, а какие будут присущи только ему.

Содержательно задания должны быть связаны с окружающим миром и диагностировать в первую очередь выделять объекты в нем, поскольку в реальном мире нас окружают различные объекты – в некотором смысле самостоятельные образования, которые обладают теми или иными параметрами и которые либо сами выполняют какие-либо действия, либо над ними можно выполнять какие-то действия. К таким объектам относятся окружающие нас предметы, животные, да и сами люди. Каждый объект, таким образом, характеризуется набором параметров и набором действий.

Так, например, объект стол обладает параметрами, определяющими его габариты, качество (тип древесины), цвет и т. д. Стол можно создать (смастерить), передвигать с места на место, на него можно ставить другие предметы, наконец, его можно уничтожить, но в любом случае он представляет собой нечто целое, пока существует.

Объекты можно классифицировать. Так, можно рассматривать класс столов – объектов, обладающих некими общими характеристиками. Каждый стол будет представителем класса столов. Этот класс и будет задавать общие характеристики всех столов. Можно пойти дальше и рассмотреть класс мебели, в который входит и класс столов. Все свойства класса мебели одновременно являются и свойствами класса столов, например, ножки, которых может быть и три, и четыре, и может быть другое количество. Однако класс столов обладает некоторыми специфическими, только ему присущими свойствами, например, стол может раздвигаться или не раздвигаться. Есть некоторые действия, которые можно выполнять с любой мебелью, в том числе и со столами. Например, мебель можно чистить. Однако разные представители мебели чистятся по-разному, например стол можно просто протереть, а мягкую мебель придется чистить с помощью пылесоса. Таким образом, для разных видов мебели придется уточнить понятие чистки, хотя оно будет всюду называться одинаково.

Отсюда можно сделать вывод, что класс столов является детализацией класса мебели, он как бы порождается классом мебели и наследует все его свойства и действия, может быть, с некоторым уточнением. В этом смысле класс столов можно считать потомком класса мебели, а класс мебели, порождающий класс столов, – его предком.

Далее можно рассмотреть класс изделий, в который будет включен класс мебели; и о классе изделий, и о классе мебели можно сделать такие же заключения, что и о классах мебели и столов. Таким образом, можно себе

представить довольно сложную иерархическую структуру «родственных» отношений классов различных объектов. [17]

Если говорить о формах диагностики, то, по нашему мнению, тестовые задания в канонической форме использовать в данном случае совершенно нецелесообразно, поскольку цель диагностики – выявить, как, каким образом протекают мыслительные процессы учащихся, а каноническая форма навязывает определенные шаблоны. Предполагается использовать задания с открытым ответом, проверяемые экспертом, в роли которого может выступить учитель.

Примеры заданий на диагностику уровня сформированности объектного мышления:

1. «Лунтик прилетел на Землю с Луны - на космическом корабле. Он очутился на поляне с цветами, грибами и ягодами, где играли Мила и Кузя.»

Ответьте на следующие вопросы:

а) Выделите присутствующие объекты в представленном тексте. (правильные варианты ответа: Лунтик, Земля, Луна, космический корабль, поляна, цветок, гриб, ягода, Мила, Кузя)

б). Опишите один из выделенных объектов – что его характеризует? Т.е., каковы его свойства? Какие действия может совершать данный объект (с данным объектом, над данным объектом?). Т.е. каковы его методы?

в) Подумайте над тем, к какому более крупному классу объектов относится данный объект, свойства и методы будут наследоваться им от родителя (являются общими для всего класса), а какие присущи только данному, конкретному объекту?

2. «Когда Румата миновал могилу святого Мики — седьмую по счету и последнюю на этой дороге, было уже совсем темно. Хваленый хамахарский жеребец оказался сущим барахлом. Он вспотел, сбил ноги и двигался скверной, вихляющейся рысью. Румата сжимал ему коленями бока, хлестал

между ушами перчаткой, но он только уныло мотал головой, не ускоряя шага. Вдоль дороги тянулись кусты, похожие в сумраке на клубы застывшего дыма. Нестерпимо звенели комары. В мутном небе дрожали редкие тусклые звезды.»

Ответьте на следующие вопросы:

а) Выделите присутствующие объекты в представленном тексте. (правильные варианты ответа: Дон Румата, могила, дорога, хамахарский жеребец, перчатка, кусты, комары, небо, звезды)

б). Опишите один из выделенных объектов – что его характеризует? Т.е., каковы его свойства? Какие действия может совершать данный объект (с данным объектом, над данным объектом?). Т.е. каковы его методы?

в) Подумайте над тем, к какому более крупному классу объектов относится данный объект, свойства и методы будут наследоваться им от родителя (являются общими для всего класса), а какие присущи только данному, конкретному объекту?

3. Отлежался-таки Данилушко. Бабушка Вихориха его на ноги поставила. Была, сказывают, старушка такая. Заместо лекаря по нашим заводам на большой славе была. Силу в травах знала: которая от зубов, которая от надсады, которая от ломоты… Ну, все как есть. Сама те травы собирала в самое время, когда какая трава полную силу имела. Из таких трав да корешков настойки готовила, отвары варила да с мазями мешала. Хорошо Данилушке у этой бабушки Вихорихи пожилось. Старушка, слышь-ко, ласковая да словоохотливая, а трав, да корешков, да цветков всяких у ней насушено да навешано по всей избе. Данилушко к травам-то любопытен – как эту зовут? где растет? какой цветок?

Ответьте на следующие вопросы:

а) Выделите присутствующие объекты в представленном тексте.

б) Опишите один из выделенных объектов – что его характеризует? Т.е., каковы его свойства?

в) Какие действия может совершать данный объект (с данным объектом, над данным объектом?). Т.е. каковы его методы?

г) Подумайте над тем, к какому более крупному классу объектов относится данный объект, свойства и методы будут наследоваться им от родителя (являются общими для всего класса), а какие присущи только данному, конкретному объекту?

Можно сделать вывод, что предварительное проведение диагностики уровня сформированности объектного мышления позволит определить, какие средства и методы обучения ООП необходимы каждому учащемуся, чтобы сделать процесс его обучения наиболее эффективным. Итоговая диагностика позволит ответить на вопрос, насколько повысился уровень его сформированности по завершению изучения тем курса «Языки и методы программирования», связанных с ООП. Важность этого момента определяется тем фактом, что согласно современным образовательным стандартам основная цель любой дисциплины – это не только формирование знаний по предмету, значительно более важны личностные результаты, связанные с развитием мышления вообще и алгоритмического мышления в частности, одной из составляющих которого и является объектное мышление.

## *2.2 Разработка ментальных карт по теме «Классы в C++» курса «Языки и методы программирования»*

Изучение объектно-ориентированного программирования вызывает определенные сложности у студентов в силу своих особенностей.

Методология ООП многогранна и нестандартна [1].Отсутствие единой, общепринятой точки зрения на объектно-ориентированный подход является как ее преимуществом, так и недостатком. Преимущество в том, что в процессе обучения можно рассматривать различные интерпретации ООП. Недостаток в том, что отсутствие единой точки зрения на ООП может привести к ее формальному и ограниченному изучению. Рассмотрение принципов ООП без обучения объектной декомпозиции, объектно-ориентированному проектированию и реализации на практике преимуществ ООП, не способствует формированию у студентов необходимых представлений об ООП.

Большую часть затруднений студенты испытывают на начальном этапе обучения: в процессе формирования представлений об основах ООП. Оттого, как формируются эти представления, будет зависеть все дальнейшее изучение методологии ООП. Традиционным способом графического представления алгоритма решения той или иной задачи на компьютере является блок-схема. Но если мы говорим не о последовательном, структурном, а об объектно-ориентированном программировании, то видим, что, во-первых блок-схемы явно недостаточно, чтобы отразить основные принципы объектно-ориентированного программирования – инкапсуляцию, наследование, полиморфизм, лежащие в основе решения задачи с использованием этой технологии, во-вторых, блок-схема, как средство обучения программированию и, следовательно, как средство формирования

алгоритмического мышления, обладает рядом недостатков, главным из которых является достаточно высокий уровень ее абстракции .

В ООП для наглядного представления структуры класса и взаимосвязи между классами существует специально созданные средства UML-диаграммы, но они, как и блок-схемы, имеют высокий уровень абстракции, и, в отличие от блок-схем, являются статичным. С их помощью невозможно отобразить алгоритмы работы с классами.

Следовательно, традиционно используемых блок-схем как средств формального представления алгоритмов решения задач и UML-диаграмм недостаточно для понимания, особенно при изучении такой сложной технологии как ООП, нужно еще одно звено – между алгоритмическим мышлением на житейском, повседневном уровне и алгоритмическим мышлением на формализованном уровне, необходимом для успешного изучения ООП. Таким недостающим звеном и принципиально новым средством обучения программированию могут служить алгоритмические ментальные карты .

В качестве примера мною разработана алгоритмическая ментальная карта, иллюстрирующая решение конкретной задачи «Создание класса Телефонная книга и работа с ним» по теме: «Классы» в С++. Для изучения этой темы студентам предлагается следующая задача: разработать программу, формирующую и обрабатывающий динамический массив объектов класса TPhoneLog. Элементы-данные: Фамилия, Имя, Отчество, Адрес, Номер, Время внутригородских разговоров, Время междугородних разговоров. Реализовать возможность вывода: а) сведения об абонентах, время внутригородских разговоров которых превышает заданное; б) сведения об абонентах, воспользовавшихся междугородней связью; в) сведения об абонентах, выведенные в алфавитном порядке. При реализации класса предусмотреть: конструкторы класса (по умолчанию; получающий параметры; получающий параметр «Ссылка на класс Т»); функции-методы

класса (ввода–вывода данных; установка значений класса; получение значений элемента-данных X класса; вывод на экран содержимого класса; методы необходимые для выполнения задания).

Для построения ментальной карты использована on-line программа Bubbl.us. Программа бесплатна, чем объясняется ее высокая востребованность пользователями. Bubbl.us позволяет редактировать графические схемы несколькими пользователями, что дает возможность организовать коллективную деятельность.

Преимущества Bubbl.us: созданную карту можно распечатать, а также поместить на сайт или в блог; над интеллект-картой может работать несколько человек одновременно; интеллект-карту можно сохранить в формате jpg ,png, html или как рисунок, а также отправить по электронной почте; каждый из членов группы имеет возможность редактировать в любое время уже созданную интеллект-карту.

Особенности Bubbl.us: нет возможности добавлять картинки; элементы интеллект-карт различаются только цветом и положением; язык интерфейса – английский.

Ментальная карта в наглядной форме демонстрирует возможный алгоритм решения поставленной задачи с момента анализа условий задачи до написания кода.

На первом уровне (рис.1) представлены основные конструкции языка С++, реализующие работу с объектами.

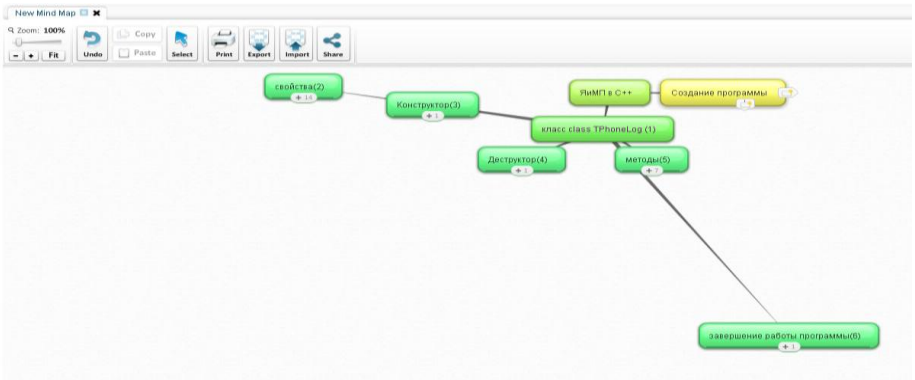



Рис.1. Первый уровень алгоритмической ментальной карты

На втором уровне (рис.2) этим конструкциям поставлено в соответствие содержание конкретной задачи

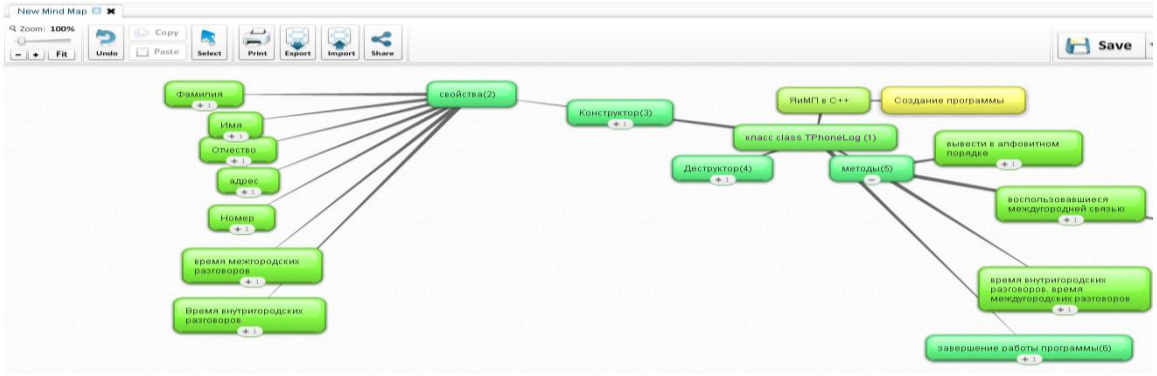



Рис.2. Второй уровень алгоритмической ментальной карты

На третьем уровне уже размещены фрагменты программного кода, реализующего решение поставленной задачи (рис.3).

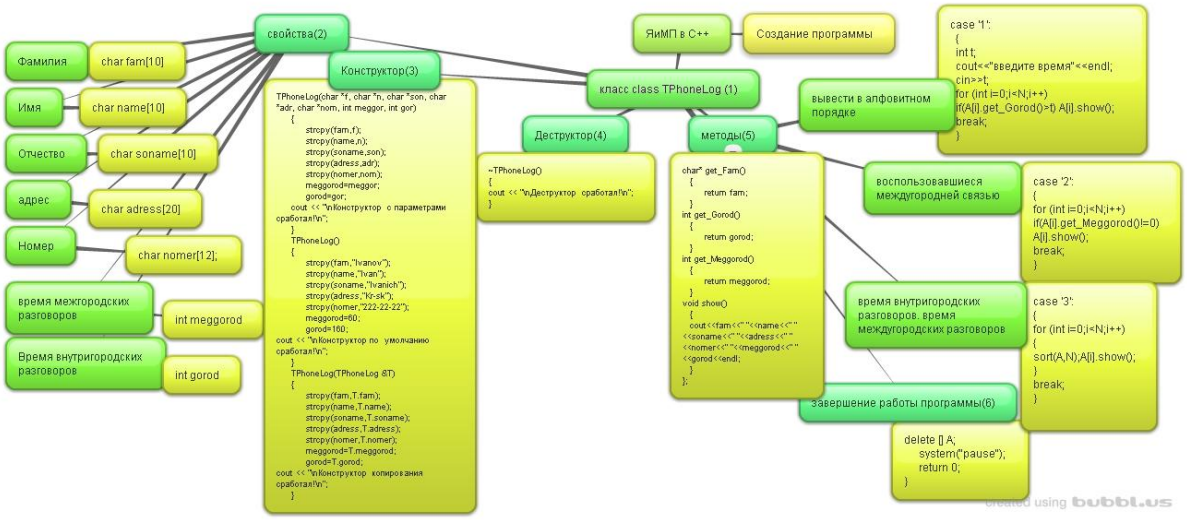



Рис.3. Третий уровень алгоритмической ментальной карты

Полностью ментальная алгоритмическая карта представлена в Приложении 2.

Аналогично была разработана ментальная алгоритмическая карта «Средний балл студента», она приведена в Приложении 3.

Представленные ментальные алгоритмические карты наглядно, с применением знаков, образов, приемов, активизирующих чувственную зону памяти, иллюстрирует мыслительные процессы, приводящая к решению задачи. Создать такую карту – это значить отразить процесс нашего мышления. Конечно же, это процесс, протекающий в мозгу объясняющего

новый материал преподавателя, но вникая в чужой мыслительный процесс, а не только видя его результат, как происходит в том случае, если объяснение протекает без поддержки ментальными картами, мыслительный процесс студентов, направленный на решение данной задачи, активизируется в нужном направлении. Кроме того, при самостоятельном решении задач на лабораторной работе, полезно предложить студентам предварительно составить ее ментальную алгоритмическую карту. Наглядно изображая процесс мышления, протекающий при решении задачи, можно увидеть пробелы, недостающую для решения задачи информацию и т.п.

### *2.3 Разработка кинестетического тренажера по теме «Классы в C++» курса «Языки и методы программирования»*

Существующие средства обучению программированию в большинстве своем нацелены на аудиальные и визуальные каналы восприятия, между тем как согласно современным подходам к теории мышления немаловажную роль в когнитивных процессах играют и моторные, тактильные ощущения. [1]

Петров А. Н. в своей статье под названием «Особенности методики обучения студентов объектно-ориентированному программированию и проектированию» [4] предлагает на начальном этапе обучения студентов ООП использовать презентации. Анимация, выделение цветом строк программного кода и соответствующих частей элементов диаграммы классов языка UML помогут студентам лучше понять взаимосвязи объектно-ориентированного программного кода и диаграммы классов языка UML. Использование презентаций на начальном этапе обучения ООП позволит студентам начать применять язык UML в объектно-ориентированном проектировании и создавать на основе него объектно-ориентированный программный код.

Однако презентации нацелены только на визуальные каналы восприятия, и они не отражают мыслительного процесса решения задачи, а только его результат. Поэтому в работах Степановой Т.А., Нигматулиной Э.А., Бархатовой Д.А. « Натурные средства обучения информатике в условиях реализации телесно-ментального подхода »[2] предлагается использовать в процессе обучения программированию ментальные карты и кинестетические тренажеры. Там же описаны данные средства обучения по школьному курсу информатики. Нам представляется актуальной разработка подобных средств и методов обучения объектно-ориентированному программированию (ООП) будущих учителей информатики.

В качестве примера нами разработан кинестетический тренажер для изучения темы «Классы в С++», согласно которому создаваемый класс представляет собой поле (например, магнитную доску) с выделенными под описание свойств класса, методов класса, конструкторов, функции, в которой осуществляется обращение к членам класса и вызов методов, областями. (рис. 17)


Рис.17. Основные конструкции языка С++, реализующие работу с объектами.

В этих областях могут быть зафиксированы в виде карманов элементы программного кода для решения конкретной задачи, заготовленные отдельно на карточках.(Рис. 18)


Рис. 18. Карточки с программным кодом.

Из непрозрачного материала заготовлены карточки, на которых приведены примеры экземпляров класса — исходные данные для решаемой задачи. (Рис. 19)


Рис.19. Карточки с исходными данными.

Размещая их в соответствующие поля, согласно свойствам созданного класса, выполняя функции, прописанные в методах класса, прослеживая, как изменяется значение той или иной переменной,. студент, по сути выполняет программу вручную и, как показывает личный опыт, досконально начинает разбираться в том, как она работает.

Для примера создан набор карточек для решения конкретной задачи по теме: «Классы» в С++. Для изучения этой темы студентам предлагается следующая задача: разработать программу, формирующую и обрабатывающий динамический массив объектов класса TPhoneLog. Элементы-данные: Фамилия, Имя, Отчество, Адрес, Номер, Время внутригородских разговоров, Время междугородних разговоров. Реализовать возможность вывода: а) сведения об абонентах, время внутригородских разговоров которых превышает заданное; б) сведения об абонентах, воспользовавшихся междугородней связью; в) сведения об абонентах, выведенные в алфавитном порядке. При реализации класса предусмотреть: конструкторы класса (по умолчанию; получающий параметры; получающий параметр «Ссылка на класс Т»); функции-методы класса (ввода–вывода данных; установка значений класса; получение значений элемента-данных Х класса; вывод на экран содержимого класса; методы необходимые для выполнения задания).

Таким образом, кинестетический тренажер для изучения темы «Классы в С++» предоставляет возможность для осуществления ручной трассировки и отладки программы. Благодаря разбиению целой программы по отдельным составляющим (конструкции языка, свойств, методов, реализации программного кода), студенты погружаются в процесс выполнения алгоритма, изучаемые понятия и методы раскрываются для них на тактильном уровне, в процессе усвоения нового материала задействуются не только модельная, понятийная, но еще и моторная зоны памяти. Что несомненно, существенно повышает уровень понимания и усвоения учебного материала.

Размещение и закрепление элементов тренажера скажем на магнитной доске, с невозможностью оперировать его отдельными частями, способствует пониманию того, что класс вместе со всеми

своими свойствами и методами представляет собой единое неделимое целое, что мы не можем оперировать ими отдельно, необходимо к каждому компоненту класса обращаться по имени класса.

Представляется, что подобные средства обучения позволят сформировать необходимый уровень объектного стиля мышления студентов, обеспечивающий высокое качество обучения объектно-ориентированному программированию.

### *2.4 Результаты апробации*

Результаты исследования были апробированы в ИМФИ КГПУ им. В.П. Астафьева среди студентов 3 курса в рамках профильного исследования.

Целью исследования являлось проверка достоверности гипотезы, а так же эффективности действия ментально-телесного подхода.

На первом занятии была проведена диагностика уровня сформированности объектного мышления при помощи разработанных нами тестовых заданий в открытой форме. Кроме того, студентам предлагалось построить ментальную карту отражающую процесс решения задачи по ООП, которая бы отображала умение разделять на объекты, методы, свойства.

Тестовые задания выглядят следующим образом:

**Ознакомиться с представленными текстами и ответить на представленные вопросы:**

**а)** Выделите присутствующие объекты в представленном тексте.

**б)** Опишите один из выделенных объектов – что его характеризует? Т.е., каковы его свойства?

**в)** Какие действия может совершать данный объект (с данным объектом, над данным объектом?). Т.е. каковы его методы?

**г)** Подумайте над тем, к какому более крупному классу объектов относится данный объект, свойства и методы будут наследоваться им от родителя (являются общими для всего класса), а какие присущи только данному, конкретному объекту?

**Текст 1.** «Лунтик прилетел на Землю с Луны - на космическом корабле. Он очутился на поляне с цветами, грибами и ягодами, где играли Мила и Кузя.»

**Текст 2.** Отлежался-таки Данилушко. Бабушка Вихориха его на ноги поставила. Была, сказывают, старушка такая. Заместо лекаря по нашим заводам на большой славе была. Силу в травах знала: которая от зубов, которая от надсады, которая от ломоты… Ну, все как есть. Сама те травы собирала в самое время, когда какая трава полную силу имела. Из таких трав да корешков настойки готовила, отвары варила да с мазями мешала. Хорошо Данилушке у этой бабушки Вихорихи пожилось. Старушка, слышько, ласковая да словоохотливая, а трав, да корешков, да цветков всяких у ней насушено да навешано по всей избе. Данилушко к травам-то любопытен – как эту зовут? где растет? какой цветок?

**Текст 3.**

«Когда Румата миновал могилу святого Мики — седьмую по счету и последнюю на этой дороге, было уже совсем темно. Хваленый хамахарский жеребец оказался сущим барахлом. Он вспотел, сбил ноги и двигался скверной, вихляющейся рысью. Румата сжимал ему коленями бока, хлестал между ушами перчаткой, но он только уныло мотал головой, не ускоряя шага. Вдоль дороги тянулись кусты, похожие в сумраке на клубы застывшего дыма. Нестерпимо звенели комары. В мутном небе дрожали редкие тусклые звезды.»

Составить ментальную карту на начальной диагностике предлагалось для решения задачи «Сложение дробей»

**Обработка результатов диагностики**

- Для каждого текста приведены 4 одинаковых задания, каждый правильный ответ а)-б) оценивается по 1 баллу, а ответы на задание в)-г) оцениваются в 2 балла. Если признак, положенный испытуемым в основу обобщения, является менее существенным и общим, чем предполагалось автором опроса, то такое задание оценивается в 1 балл для а) и б), и 2 балла

для в) и г). Если признак, являющийся основой обобщения, найден к меньшей половине , то выполнение задания оценивается в 1 балл для в) и г). Если признак, являющийся основой обобщения, найден неверно (или не найден совсем), то выполнение задания оценивается в 0 баллов.

- Общий балл, представляющий собой оценку за диагностику уровня сформированности объектного мышления, подсчитывается путем суммирования баллов, полученных испытуемым за выполнение каждого из 4 вопросов. Максимальный балл 18, а минимальный - 3.

Кроме того, оценивалось составление ментальной карты по уровням ее детализации от 3 до 9 баллов.

Определение уровней проходит по следующим критериям**:**

- Высокий уровень сформированности объектного стиля мышления— более 23 баллов;
- Средний уровень сформированности объектного стиля мышления— от 12 до 23 баллов
- Низкий уровень сформированности объектного стиля мышления — от 6 до 11 баллов.

На следующем занятии студентам объясняется задача «Создание класса Телефонная книга и работа с ним» по теме «Классы в С++». Студентам, у которых был выявлен низкий уровень ОСМ предлагаются кинестетический тренажер и ментальная карта по данной теме. Студентам, у которых средний уровень ОСМ предлагается ментальная карта. А оставшиеся студенты, которые показали высокий уровень ОСМ записывают код сразу в С++

На последнем занятии студентам предлагается повторно выполнить те же тестовые задания, но с другими текстами и составить ментальную карту по решению задачи «Успеваемость студентов» по теме «Классы в С++».

Начальные результаты диагностики уровня сформированности ОСМ и уровня усвоения предметных знаний сопоставляются с результатами диагностики, проводимой по окончании наших занятий.

Итоги обработки результатов представлены на рис. 20.

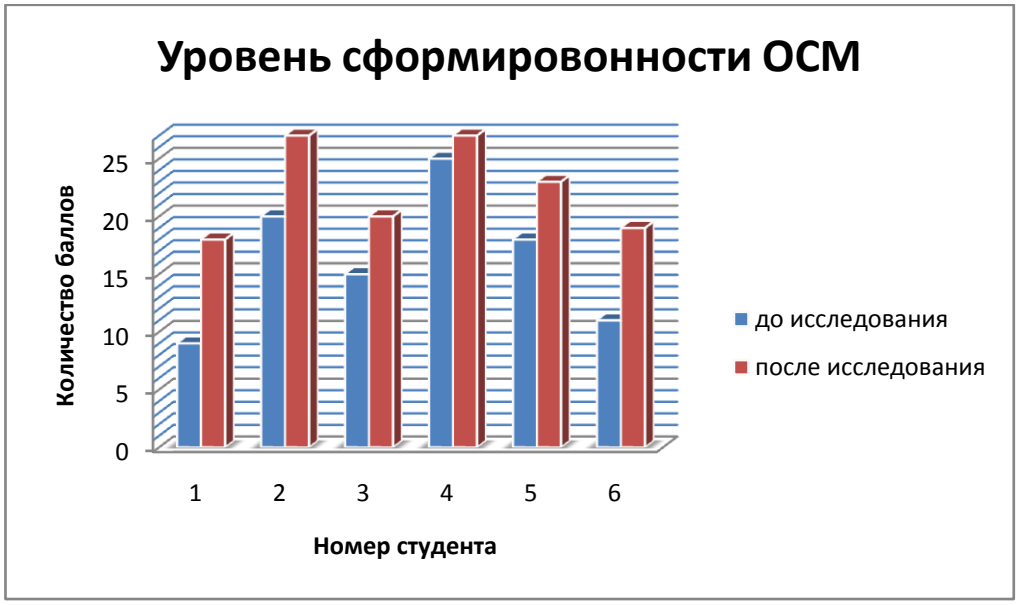



Рис. 20. Уровень сформировонности ОСМ у студентов до и после исследования.

Полученные результаты позволяют заключить, что применение разработанных нами средств обучения на основе ментально-телесного подхода способствует повышению уровня сформированности объектного стиля мышления и эффективно влияет на уровень понимания объектно-ориентированного программирования на языке С++, что говорит о справедливости выдвинутой нами гипотезы.

***Выводы по 2 главе***

В соответствии с поставленной целью дипломной работы и для более глубокого анализа процесса формирования объектного мышления студентов, определения его уровней сформированности, подобраны задания для диагностики уровня сформированности объектного мышления. Результаты апробации позволяют утверждать, что с помощью данной диагностики можно адекватно оценить уровень развития объектного мышления у студентов.

На основе проведенной диагностики можно дифференцировать студентов по уровню сформированности объектного мышления и в зависимости от этого подбирать наиболее подходящие для каждого средства обучения, способствующих развитию объектного мышления. Одним из таких средств являются ментальные карты. Приводится описание разработанных ментальных карт и кинестетического тренажера по теме: «Классы в С++» курса «Языки и методы программирования» и результаты их апробации.

Разработанные средства могут быть использованы в учебном процессе по курсу «Языки и методы программирования».

## ЗАКЛЮЧЕНИЕ

В результате проведенного исследования достигнута цель, выполнены все поставленные задачи:

– Изучена сущность ментального подхода к обучению, определено, что ментально-телесный подход предполагает смещение целеполагания учебного процесса в сторону развития когнитивных способностей обучаемых, в частности, если рассматривать обучение ООП – то основной целью будет являться развитие объектного стиля мышления. В рамках ментально-телесного подхода к обучению программированию предполагается использование методики ментальных карт и кинестетических тренажеров как эффективных средств развития алгоритмического мышления, нацеленных на когнитивные особенности студентов

– Описано возникновение методологии ООП, его основные принципы, UML-диаграммы как средство визуализации в ООП

– Выделены особенности обучения объектно-ориентированному программирования в педагогическом вузе, заключающиеся в том, что современные образовательные стандарты предъявляют высокие требования к предметной подготовке учителя. Учитель информатики должен владеть всеми современными технологиями программирования, поэтому в курс «Языки и методы программирования» в педагогическом вузе включено изучение не только императивного программирования, которое изучается в школе, но и логического, функционального, объектно-ориентированного, параллельного и др. современных технологий программирования. Следовательно, алгоритмическое мышление будущего учителя информатики должно быть развито на самом высоком, профессиональном уровне. Кроме того, оно должно быть расширено еще и методическим компонентом, поскольку у педагогов не только у самих должно быть сформировано алгоритмическое мышление на самом высоком уровне, но

они еще должны быть способными формировать и развивать алгоритмическое мышление своих учеников.

– Выявлено, что на успешность обучения объектно- ориентированному программированию влияет уровень сформированности объектного стиля мышления.

– Уточнено понятие объектного мышления, построена его структурная модель..

– В соответствии с поставленной целью дипломной работы и для более глубокого анализа процесса формирования объектного мышления школьников и студентов, определения его уровней сформированности, разработаны задания для проведения диагностики уровня сформированности объектного мышления.

– Определено, что на основе проведенной диагностики можно дифференцировать студентов по уровню сформированности объектного мышления и в зависимости от этого подбирать наиболее подходящие для каждого средства обучения, способствующих развитию объектного мышления.

– Разработаны ментальные карты «Телефонная книга» и «Средний балл студенты» по теме «Классы» в C++. Особая необходимость использования подобных ментальных карт при объяснении того, как тот или иной алгоритм можно реализовать на языке объектного-ориентированного программирования видится в том, что в силу особенностей объектно-ориентированного программирования применение традиционных блок-схем, UML-диаграмм не всегда приводит у ожидаемым результатам.

– Разработан кинестетический тренажер по теме «Классы» в C++. Кинестетические тренажеры являются принципиально новым средством обучения, нацеленным, в отличие от сущестующих не только на аудиальные и визуальные, но и на кинестетические каналы восприятии

Перспективы дальнейших исследований в данном направлении видятся следующие:

- - Совершенствование системы диагностики уровня сформированности ОСМ
- - Дальнейшая доработка средств развития ОСМ – ментальных карт и кинестетических тренажеров
- - Использование технологий 3D-печати для создания кинестетических тренажеров

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Joseph D. Novak, Cornell University. The Theory Underlying Concept Maps and How To Construct Them. – http://cmap.coginst.uwf.edu/info/index.html
2. Алюшин А.Л., Князева Е.Н. Телесный подход в когнитивной науке. - Философские науки. - № 2, 2009.
3. Бертран М. Объектно-ориентированное конструирование программных систем / Пер. с англ. - М.: Издательско-торговый дом «Русская Редакция», 2005
4. Бертран М. Объектно-ориентированное конструирование программных систем / Пер. с англ. - М.: Издательско-торговый дом «Русская Редакция», 2005
5. Бьюзена Т. Суперпамять. Издательство «Попурри» ,2008г.
6. Бьюзена Т. Суперпамять. Издательство «Попурри» ,2008г.
7. Вирт Н. Алгоритмы + структуры данных = программы: Пер. с англ.— М.: Мир,1985.—406с
8. Вирт Н. Алгоритмы + структуры данных = программы: Пер. с англ.— М.: Мир,1985.—406с
9. Газейкина А.И. Стили мышления и обучение программированию студентов педагогического вуза. URL: http://ito.edu.ru/2006/Moscow/Lhtml
10. Газейкина А.И. Стили мышления и обучение программированию студентов педагогического вуза. URL: http://ito.edu.ru/2006/moscow(дата обращения: 14.12.2010).
11. Голубцова А.В., Грук Е.Д., Степанова Т.А. Разработка ментальных алгоритмических карт по теме «Основные алгоритмические

структуры» // Материалы II Всероссийской научно-практической конференции с международным участием «Перспективы и вызовы информационного общества», 2013

12. Гребнева Д.М. Обучение школьников программированию на основе семиотического подхода: Автореферат дис. ... канд. пед. наук. - Екатеринбург, 2014. - 24 с.
13. Гребнева Д.М. Обучение школьников программированию на основе семиотического подхода: Автореферат дис. ... канд. пед. наук. - Екатеринбург, 2014. - 24 с.
14. Жужжалов В.Е. Интеграционные методы изучения программирования в вузовском курсе информатики // Вестник МГПУ, Серия “Информатика и информатизация образования”, М., 2003, № 1 (1).
15. Жужжалов В.Е. Интеграционные методы изучения программирования в вузовском курсе информатики // Вестник МГПУ, Серия “Информатика и информатизация образования”, М., 2003, № 1 (1).
16. Жужжалов В.Е. Основы интеграции парадигм программирования в курсе информатики // Российская академия образования. Институт содержания и методов обучения.- М.: Образование и информатика, 2004. – 127 с.
17. Жужжалов В.Е. Основы интеграции парадигм программирования в курсе информатики // Российская академия образования. Институт содержания и методов обучения.- М.: Образование и информатика, 2004. – 127 с.
18. Загвязинский В.И. Теория обучения: современная интерпретация: учеб. пособие для студентов высш. пед. учеб. заведений / В.И. Загвязинский. – М.: Академия, 2001. – 192 с.
19. Загвязинский В.И. Теория обучения: современная интерпретация: учеб. пособие для студентов высш. пед. учеб. заведений / В.И. Загвязинский. – М.: Академия, 2001. – 192 с.

20. Иванова Г.С. Объектно-ориентированное программирования: учеб. для вузов / Г.С. Иванова, Т.Н. Ничушкина, Е.К. Пугачев; под ред. Г.С. Ивановой. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 320 с.

21. Кауфман, В.Ш. Языки программирования: концепции и принципы / В.Ш Кауфман. – М. : ДМК-пресс, 2010. – 464 с.

22. Кауфман, В.Ш. Языки программирования: концепции и принципы / В.Ш Кауфман. – М. : ДМК-пресс, 2010. – 464 с.

23. Киргизова Е.В. Методика обучения студентов теоретической информатике на информационно-деятельностной основе. Дис. … канд. пед. наук. – Красноярск, 2010. – 201 с.

24. Киргизова Е.В. Методика обучения студентов теоретической информатике на информационно-деятельностной основе. Дис. … канд. пед. наук. – Красноярск, 2010. – 201 с.

25. Крюков В.А. Анализ принципов объектно-ориентированного программирования // Микропроцессорные средства и системы, № 2, 1989

26. Леонтьев А.Н. Деятельность. Сознание. Личность. М.: Смысл; Издательский центр «Академия», 2004. 352 с.

27. Марченко Л. Разработка ментальной карты для изучения темы: «Классы» в С++// Сборник научных трудов по материалам международной научно- практической конференции. «Вопросы образования и науки: теоретический и методический аспекты» Тамбов, 30 июня, 2015 год

28. Марченко Л.С. Диагностика сформированности объектного мышления//Международная научно – практическая конференция «Вопросы образования и науки» Тамбов, 31 декабря, 2015 год

29. Марченко Л.С. Использование методики ментальных карт при обучении объектно-ориентированному программированию в

педагогическом вузе// XVI Всероссийский (с международным участием) научно-практический форум студентов, аспирантов, и молодых ученых. «Молодежь и наука XXI» Касноярск, 19-20 мая, 2015 год

30. Марченко Л.С. Ментальный подход к обучению объектно-ориентированного программирования //Актуальные проблемы информатики и информационных технологий в образовании: материалы Всероссийской научно-практической конференции с международным участием в рамках XVI международного научно-практического форума студентов, аспирантов и молодых ученых «Молодежь и наука XXI века». Красноярск, 17 мая 2016 г. [Электронный ресурс] / ред. кол.; отв. ред. П.С. Ломаско. – Электрон. дан. / Краснояр. гос. пед. ун-т им. В.П. Астафьева. – Красноярск, 2016. – 159 с. URL: http://elib.kspu.ru/document/17540

31. Марченко Л.С. Особенности изучения объектно- ориентированному программированию в педвузе// Международная научно – практическая конференция «Актуальные вопросы в научной работе и образовательной деятельности» Тамбов, 30 мая, 2015 год

32. Марченко Л.С. Уточнение понятий «объектное мышление» на основе информационного// «Информация и образование: границы коммуникаций» INFO'15 Information and education: borders of communication, Горго-Алтайск, Республика Алтай, 5-8 июля, 2015

**33.** Нигматулина Э.А., Сокольская М.А., Степанова Т.А. Расширение понятия алгоритмического мышления при изучении современных технологий программирования в педагогическом вузе (статья) // Материалы VIII Международной научно-практической конференции «Педагогический профессионализм в образовании». – Новосибирск, 2012. – с.152-158.

**34.** Нигматулина Э.А., Степанова Т.А. Условия формирования алгоритмической культуры студентов на основе информационного подхода // Вестник Красноярского государственного педагогического университета им. В.П.Астафьева. 2011 (1) / Красноярский государственный педагогический университет им. В.П. Астафьева. – Красноярск, 2011. – 280 с. [с. 82-86] (Журнал из перечня ВАК).

35. Степанов М.А. Опыт мышления тела: автореф. дис. … канд. философ.наук. СПб, 2011. с.6

36. Степанова Т.А. Теория алгоритмического мышления. Краснояр. гос. пед. ун-т им. В.П. Астафьева. Красноярск, 2014

37. Турский М. Методология программирования. М., Мир, 1981.

38. Тхостов А.Ш. Психология телесности. — М.: Смысл, 2002. – 287 с.

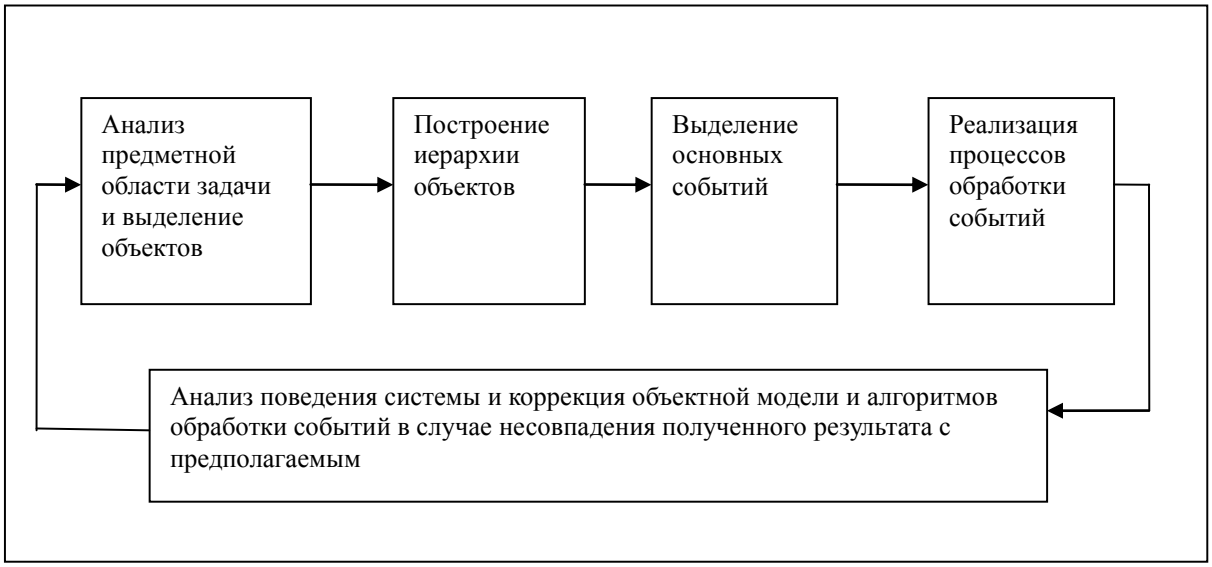
39. Хорев П.Б. Технологии объектно-ориентированного программирования: Учеб. пособие для студ. высш. учеб. заведений / П.Б. Хорев. – М.: Издательский центр «Академия», 2004. – 448 с.

40. Э.Нигматулина, Н.И.Пак, М.А.Сокольская, Т.А.Степанова Программирование//2т. Т1: учебник для студ.учреждений высш.проф.образование /:под ред.Н.И.Пак-М.:издательский центр «Академия» 2013г

ПРИЛОЖЕНИЯ 1

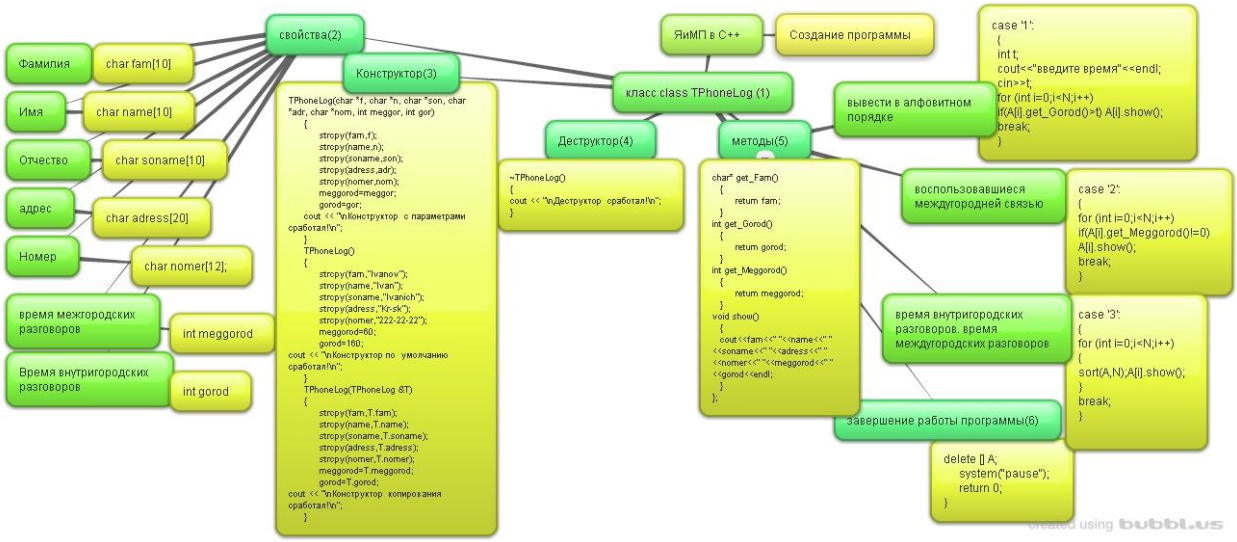
I Объектное мышление

Структурная модель объектного стиля мышления



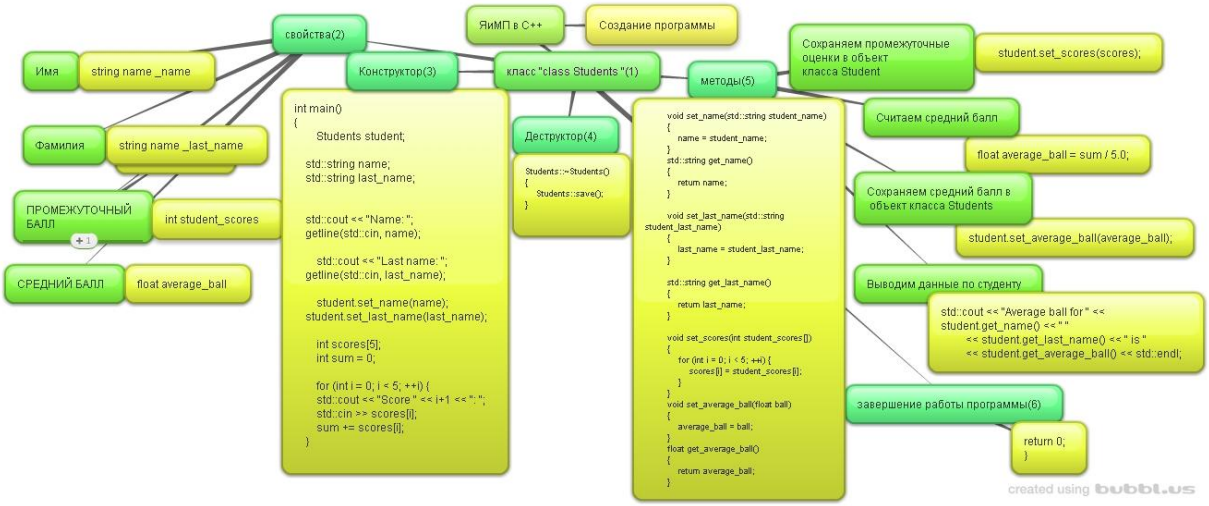
Приложение 2.

Алгоритмическая ментальная карта «Телефонная книга» по теме «Классы в С++» курса «Языки и методы программирования»



Приложение 3.

Алгоритмическая ментальная карта «Учет успеваемости студентов» по теме «Классы в С++» «Языки и методы программирования»



Приложение 4.

Материалы для диагностики уровня сформированности объектного стиля мышления

1.Задание на диагностику уровня сформированности ОСМ.

**Ознакомиться с представленными текстами и ответить на представленные вопросы:**

**а)** Выделите присутствующие объекты в представленном тексте.

**б)** Опишите один из выделенных объектов – что его характеризует? Т.е., каковы его свойства?

**в)** Какие действия может совершать данный объект (с данным объектом, над данным объектом?). Т.е. каковы его методы?

**г)** Подумайте над тем, к какому более крупному классу объектов относится данный объект, свойства и методы будут наследоваться им от родителя (являются общими для всего класса), а какие присущи только данному, конкретному объекту?

**Текст 1.** «Лунтик прилетел на Землю с Луны - на космическом корабле. Он очутился на поляне с цветами, грибами и ягодами, где играли Мила и Кузя.»

**Текст 2.** Отлежался-таки Данилушко. Бабушка Вихориха его на ноги поставила. Была, сказывают, старушка такая. Заместо лекаря по нашим заводам на большой славе была. Силу в травах знала: которая от зубов, которая от надсады, которая от ломоты… Ну, все как есть. Сама те травы собирала в самое время, когда какая трава полную силу имела. Из таких трав да корешков настойки готовила, отвары варила да с мазями мешала. Хорошо Данилушке у этой бабушки Вихорихи пожилось. Старушка, слышь-ко, ласковая да словоохотливая, а трав, да корешков, да цветков всяких у ней

насушено да навешано по всей избе. Данилушко к травам-то любопытен – как эту зовут? где растет? какой цветок?

**Текст 3.**

«Когда Румата миновал могилу святого Мики — седьмую по счету и последнюю на этой дороге, было уже совсем темно. Хваленый хамахарский жеребец оказался сущим барахлом. Он вспотел, сбил ноги и двигался скверной, вихляющейся рысью. Румата сжимал ему коленями бока, хлестал между ушами перчаткой, но он только уныло мотал головой, не ускоряя шага. Вдоль дороги тянулись кусты, похожие в сумраке на клубы застывшего дыма. Нестерпимо звенели комары. В мутном небе дрожали редкие тусклые звезды.»

Приложение 5.

Кинестетический тренажер  по теме «Классы в С++».