МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

УТВЕРЖДАЮ
Председатель совета
_____
«__»_____2014 г.

Программа дисциплины

«Современные методы программирования (Программирование 2)»

Образовательная программа: прикладная математика и информатика — 010400

Квалификация (степень) выпускника
Бакалавр

Форма обучения
Очная

Новосибирск
2014

Программа учебной дисциплины
«Современные методы программирования (Программирование 2)»

Курс является продолжением курса Программирование 1, который изучается на первом курсе ММФ НГУ и преподается студентам ММФ в первом семестре второго курса. Он разделяется на две части. Первая из них имеет целью получение навыков программирования на C++, вторая посвящена знакомству с различными стилями программирования и их методами. Знания и навыки, получаемые в рамках преподавания курса, необходимы всем, кто намерен уметь осознано пользоваться вычислительными и программными средствами. Курс способствует развитию мышления и организованности обучаемых. Также он создает понятийную базу, необходимую для профессиональных программистов всех специальностей.

Формы обучения курса — лекционные и семинарские занятия, а также самостоятельная работа студентов.

Автор: И.Н. Скопин, к.ф.-м.н., с.н.с., доцент
Факультет: Механико-математический
Кафедра: Программирование

1. Цели и задачи освоения дисциплины (курса)

Целью курса является обеспечение для студентов возможности развития навыков программирования, полученных при изучении курса «Программирование 1». Кроме знания и навыком программирования на C++, обучаемые получают общее представление о стилях программирования и сферах их применения, о методах этих стилей. В результате освоения курса студенты становятся подготовлены для самостоятельного развития полученных навыков и знаний. Особое внимание уделено указанию границ применения излагаемых методов.

Курс предназначен для студентов 2 курса ММФ НГУ различных направлений специализации при дальнейшем обучении в университете.

Актуальность курса

Курс нацелен на подготовку специалистов, которые будут продуктивно использовать вычислительные и программные системы, а также развивать их в дальнейшем. С учетом того, что сегодня применение информационных технологий является повсеместным, что программистский образ мышления необходим для современных научных исследований, актуальность тематики курса представляется очевидной.

Учебные задачи курса подразделяются на частные и общие, что соответствует двум аспектам предлагаемого материала: преподаванию C++ и методов, связанных с различными стилями программирования.

Задачи, связанные с C++:

1. Получить представление, необходимое и достаточное для практического использования о базовых понятиях и принципах объектно-ориентированного программирования;
2. Дать информацию о метамодели C++ и ее влиянии на практические методы программирования;
3. Показать модели памяти и вычислений C++, подчеркивая их общность для объектно-ориентированного программирования и детали, связанные со спецификой языка;
4. Разъяснить способы использования на практике общих и специальных средств и инструментов программирования на C++;

5. Дать сведения, необходимые для продуктивного использования стандартных библиотек.

Задачи, связанные с изучением стилей программирования:

1. Дать представление о моделях вычислений языков и их связи со стилями программирования;
2. Показать взаимосвязь стилей и методов программирования;
3. Дать обзор методов и языковых средств сентенциального стиля программирования
4. Дать обзор методов и языковых средств функционального стиля программирования.

2. Место дисциплины в структуре образовательной программы

Курс базируется на математических курсах первого года обучения на ММФ НГУ. Знания других предметов для его освоения не требуется.

Дисциплина курса рассматривается как инструментальная, способствующая усвоению других предметов, преподаваемых в ММФ НГУ, например, курса вычислительных методов. В то же время, для тех, кто намерен специализироваться в области дискретной математики, информатики, системного или прикладного программирования, материал курса является базой для дальнейшего обучения и самостоятельной работы.

3. Компетенции обучающегося, формируемые в результате освоения дисциплины

Процесс изучения дисциплины направлен на формирование у выпускников следующих компетенций:

- общекультурные компетенции: ОК-1, ОК-2, ОК-5, ОК-6, ОК-8, ОК-9, ОК-11;
- профессиональные компетенции: ПК-2, ПК-3, ПК-7, ПК-8, ПК-15 , ПК-21, ПК-25 , ПК-27.

В результате освоения дисциплины обучающийся должен:

- ***Знать*:**
  - Средства объектно-ориентированного программирования на языке C++;
  - Специфические средства поддержки и методы объектно-ориентированного стиля программирования в C++;
  - Основные средства и методы функционального стиля программирования;
  - Модель разработки программ, которая соответствует концепции Model-View-Controller.
- ***Уметь*:**
  - Применять знания средств программирования на языке C++ для разработки программ среднего уровня сложности;
  - Использовать для решения программистских задач адекватные им средства;
  - Использовать доступные средства для реализации методов, адекватных решаемой задачи;
  - Разбивать задачи на подзадачи для упрощения и оптимизации решения.

4. Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 130 часов, или 8 зачетных единиц ЗЕ (одной ЗЕ соответствует 36 академических часов). Структура курса, его содержание, а также трудоемкость по видам учебной работы представлены в Таблице 1.

Таблица 1

Структура содержание виды учебной работы приобретаемые компетенции курса «Руководство командой и управление инновационными наукоемкими программными проектами»

Раздел дисциплины	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость *(в академических часах)*						Формы текущего контроля успеваемости *(по неделям семестра)* Форма промежуточной аттестации *(по семестрам)*
			Лекции	Семинары	Самостоятельная работа	Экзамен	Всего занятий + самостоятельной работы	Зачетные единицы (ЗЕ)	
1	2	3	4	5	6	7	8	9	10
Введение. Модель вычислений фон Неймана и традиционные языки. Эволюция языков программирования. Поколения языков программирования. Понятие стилей программирования и методов, соответствующих стилям.	3	1	2				**2+0**		
1. Стили и методы программирования									
1.1. **Нетрадиционные модели вычислений** Повелительное и изъявительное наклонения. Системы продукций. Системы функций. Коммутационные системы. Ассоциативные системы. Аксиоматические системы. Модели вычислений и стили программирования. *Задание для самостоятельной работы*: Реферат на тему одной из нетрадиционных моделей вычислений	3	3	2		20				Контрольная работа (5 минут)
1.2. **Функциональное программирование** Ленивые вычисления и функциональная модель. Постулаты необходимости, их следствия. Особенности ленивых и жадных вычислений при решении различных задач. Решение численных задач в функциональном стиле. Ленивые вычисления: императивные примеры	3	5	2						Контрольная работа (5 минут)
1.3. **Сентенциальное программирование** Элементы сентенциального стиля. Синтаксический анализ и вычисление предложения. Сопоставление с образцом.	3	7	2						Контрольная работа (5 минут)
1.3.1. **Язык Prolog** Логический вывод утверждений на основе фактов и язык Prolog. Семантика и стратегия вычислений Prolog'а. Базы знаний в Prolog'е.	3	9	2						Контрольная работа (5 минут)
1.3.2. **Язык Рефал** Алгорифм Маркова. Определение языка Рефал. Проецирование. Разрешение неоднозначностей. Дополнительные возможности. Внешние вычисления в Рефале.	3	11	2						Контрольная работа (5 минут)
1.4. **Концепция «Model View Controller» (MVC)** Система и ее декомпозиция, моделирование. Понятия MVC. Когда применять MVC?	3	13	2						Контрольная работа (5 минут)
Итоговая проверка знаний по первой части	3	14	2						Итоговая контрольная работа
Всего по первой части:			14		20		**16+20**	**1**	

Продолжение таблицы 1

1	2	3	4	5	6	7	8	9	10
2. Программирование на С++									
2.1. История развития С++. Проблемы совместимости с С. *Задание для самостоятельной работы* № 1	3	1		2	20				
2.2. Типы данных С++, конструируемые типы. Область определения и видимости. Перегрузка функций.	3	1		2					
2.3. Процедурные средства С++. Указатели и константы, Ссылки. Объекты. Классификация методов объекта *Задание для самостоятельной работы* № 2	3	2		2	20				
2.4. **Принципы ООП. Метамодель. Классы**									
2.4.1. Объектная модель (Абстрагирование, инкапсуляция, модульность, иерархия. типизация, параллелизм). Понятие класса. Структура класса. Управление памятью.	3	3		2					
2.4.2. Отношения между объектами (композиция, агрегация). Жизненный цикл объектов. Операторы. Перегрузка операторов.	3	4		2					
2.4.3. Средства модульности. Этапы трансляции (компиляция, компоновка, запуск и завер-шение работы программы). Заголовочные файлы («.h», «.hpp») *Задание для самостоятельной работы* № 3	3	5		2	15				
2.4.4. Иерархии классов. Наследование (уточнение, расширение, переиспользование). Виртуальный полиморфизм (виртуальная функция). Спецификаторы доступа к членам класса.	3	6		2					
2.5. **Инструментальные средства языка С++**									
2.5.1. Исключения. Перехват исключений. Исключения стандартной библиотеки. Декларация списка исключений. Обработка исключений в конструкторе.	3	7		2					Контрольная работа.
2.5.2. Множественное наследование. Абстрактные классы и интерфейсы. Динамическое приведение типов *Задание для самостоятельной работы* № 4	3	8		2	15				
2.5.3. Шаблоны. Полиморфизм классов	3	9		2					
2.5.4. Шаблоны функций, специализация. Правила разрешения перегрузки.	3	10		2					
2.6. **Стандартная библиотека**		11							
2.6.1. Аллокаторы, Итераторы. Обобщенные алгоритмы. Контейнеры STL *Задание для самостоятельной работы* № 5	3	12		2	10				
2.6.2. Новые возможности С++. Ссылки на временные объекты и перенос данных. Статическая диагностика, Внешнее инстанцирование шаблонов. Анонимные функции. explicit операторы преобразования типа	3	13		2					Контрольная работа.
Проверка и прием самостоятельных работ	3	14		2					Дифференцированный зачет
Всего по второй части:				32	80		32+80	3	
Итого по курсу:			**16**	**32**	**64**	**0**	**48+100**	**4**	

5. Образовательные технологии

В преподавании курса реализуется подход к обучению студентов, основанный на широком применении наряду с традиционными обучающими технологиями и формами обучения (чтение лекций и проведение обычных семинарских занятий) активных методов обучения, соответствующих современной образовательной парадигме обучения в деятельности. Исходя из опыта обучения языкам программирования, непосредственное преподавания методов программирования на С++ перенесено на семинарские занятия в ком-

пьютерных классах, где студенты самостоятельно выполняют задания, связанные с построением и отладкой практических программ. Содержание самостоятельной работы, связанной с теоретической частью курса, которая изучается на лекциях, заключается в подготовке рефератов по тематике дисциплины. Консультации по курсу учебным планом не регламентируются. Они проводятся в форме ответов на вопросы студентов и обсуждений.

6. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации и по итогам освоения дисциплины

Текущая оценка студента складывается из баллов, набранных на самостоятельных и контрольных работах по С++, оценки реферата и оценок пятиминутных контрольных работ, которые проводятся в начале каждой лекции с целью выяснения степени усвоения материала предыдущей лекции. и итоговой контрольной работы.

Итоговый контроль освоения материала курса — дифференцированный зачет, который включает итоговую контрольную работу, проводимую в письменной форме и включает ответы на 2 вопроса по тематике курса, решение тестов и задач. Итоговая оценка выставляется путем суммирования баллов, набранных в ходе текущего контроля и итоговой контрольной работы. Минимальное зачетное число баллов, выставляемых по результатам текущего и итогового контроля — 70 баллов. Число баллов на оценку отлично — 100 баллов.

В таблице 2 приводится количество баллов, которое можно набрать по каждой составляющей контроля.

Таблица 2

Структура оценивания деятельности обучаемых

Вид контролируемых работ	Число работ	Максимальная оценка за одну работу	Суммарная оценка
Самостоятельная работа по С++	5	12	60
Реферативная самостоятельная работа	1	8	8
Контрольная работа по С++	2	5	10
Пятиминутная контрольная работа	6	**2**	12
Итоговая контрольная работа	1	10	10
Итого:	**15**	**—**	**100**

6.1.Примеры задач для самостоятельных работ:

***Самостоятельная работа* №1**

Задание 1. Написать программу, которая заменяет в текстовом файле все вхождения какой-либо строки на другую строку, и выводит результат в новый файл.

Предполагается, что исходный файл содержит строки длиной не более 256 символов (включая знаки разделителя строки). Исходный файл задается параметром командной строки и читается построчно. Для замены содержимого строк воспользуйтесь классом стандартной библиотеки *std::string.*

Аргументы командной строки программы:

```
> changer.exe input.txt output.txt Hello Goodbye
```

Требования к программе:

1. Работа с файлами должна осуществляться стандартными средствами С (*fscanf, fprintf*).
2. Каждая прочитанная строчка должна быть преобразована в тип *std::string*

3. Для замены содержимого строки воспользуйтесь методами класса *std::string* (см. http://www.cplusplus.com/reference/string/string/)
4. При записи нового файла тип *std::string* можно приводить к С-строкам (см. документацию к классу *std::string*)
5. Преобразование исходного файла должно быть вынесено в отдельный модуль и реализовано в виде функции *replace_strings(FILE *in, FILE *out, char *what, char *repl);*
6. Функция *replace_strings* должна быть объявлена в отдельном пространстве имен.

***Самостоятельная работа* №2**

Задание 1. Написать программу сортировки содержимого файла на диске.

Исходный файл содержит данные одного из 3 типов: строки либо целые числа либо вещественные числа. Тип содержимого задается ключом в первом параметре программы. Максимальная длина строки в файле — 1024 символа. Строки могут содержать пробелы и разделены символом первода строки '\n'. Режим сортировки (по возрастанию, по убыванию) задается ключом во втором параметре программы.

Исходный файл задается третьим параметром, файл для сохранения результата — четвертым параметром программы:

filesort.exe -x -y input_file output_file

Где **x** может принимать значения: **s** (строки), **i** (целые числа), **f** (плавающая точка)
Где **y** может принимать значения: **a** (ascending — по возрастанию), **d** (descending — по убыванию)

Общие указания:

1. Для чтения исходного файла нужно использовать функцию scanf
2. Исходные данные читать из файла построчно в буфер, после чего прочитанную строку (массив символов) сохранять в динамической памяти выделяемой с помощью malloc, а указатель на строку сохранять в структуре (динамическом массиве) изменяемого размера.

***Самостоятельная работа* №3**

Задание 1.

Написать программу реализующую работу с упрощенным григорианским календарем (датами). Использовать следующие правила для календаря:

1. Минимальный год — 1-ый год н.э., максимальный год 9999 н.э.
2. Момент времени представляется годом, месяцем, числом месяца, часом, минутами, секундами
3. Каждый год, номер которого кратен 4 является високосным
4. Каждый год, номер которого кратен 100 не является високосным
5. Каждый год, номер которого кратен 400 является високосным
6. Часовой пояс не учитывается (считается равным UTC)
7. Не учитывать правило, согласно которому следующим днем после 4 октября 1582 года идет 15 октября 1582 года
8. Не требуется совпадение юлианского и данного календаря в датах до 4 октября 1582 года (проблема високосных годов кратных 100 но не кратных 400)

Класс даты должен поддерживать корректные операции приращения минут, секунд, часов, дней, месяцев и лет с учетом правил календаря.

Месяцы реализовать перечислением Month {Jan=1, …, Dec = 12}.

Конструктор по умолчанию должен создавать дату соответствующую текущему времени UTC (для получения времени можно использовать функции из time.h).

Класс должен реализовывать конструктор копии.

Класс должен обеспечить полный набор методов селекторов для получения текущего года, месяца, дня месяца (числа), часа, минут, секунд.

Реализовать полный конструктор (unsigned аргументы — год, месяц, число, час, минуты, секунды).

В случае если значения в конструктор переданы некорректно, осуществлять автоматическую нормализацию даты (значение большее допустимого в определенном поле приводит к увеличению старших полей, таким образом вызов конструктора от аргументов Date(2010, Feb, 30, 22, 59, 72) приведет к созданию объекта соответствующего Date(2010, Mar, 2, 23, 00, 13))

Реализовать дополнительно конструкторы:

- Date (unsigned int year, Month m, unigned int day) — создает объект с временем 0 ч 0 м 0 сек
- Date (unsigned int hrs, unsigned int mnts, unsigned int days) - создает объект с указанным временем и текущей датой

Операции addYears, addMonths, addDays, addHours, addMinutes, addSeconds должны возвращать новый объект «по значению», оставляя исходный объект неизменным.
Класс должен реализовывать оператор присваивания (единственный метод-модификатор в классе). Таким образом для изменения исходного объекта нужно пользоваться конструкцией вида:

Date theDate = theDate.addMonths(1);

Аргументы методов addXXX могут быть отрицательными, но при этом дата не может быть меньше 0:00:00 1 Jan 1 или больше 9999-12-31 23:59:59

Реализовать метод std::string toString() const возвращающий строковое представление даты в формате:

YYYY-MMM-DD hh::mm::ss

Написать тестовую программу, демонстрирующую корректную работу календаря.

Самостоятельная работа №4

Задание 1. Реализовать класс Vector для вещественных чисел. Вектор должен хранить массив вещественных чисел выделенных в свободной памяти (по new).

В случае переданного некорректного аргумента размера или индекса просто выбрасывать std::out_of_range исключение, никакой дополнительно обработки не делать (прямым throw std::out_of_range ("Illegal size") либо std::out_of_range("out of bounds")).

Индексация элементов ведется с нуля (при использовании оператора [ ]) .

Требования к реализации вектора:

1. Набор конструкторов
    a. explicit Vector(n); //конструктор с резервированием памяти. инициализирован нулями
    b. Vector (n, const double* dp); //конструктор от массива double и с заданным размером, данные для вектора должны копироваться из массива
    c. Vector (const Vector &); //конструктор копии
    d. Vector (n, const Vector& another) // конструктор создающий вектор размера n и заполняющий его элементами из другого вектора, при необходимости подставляя вместо недостающих нули, либо отбрасывающий лишние
2. ~Vector() //деструктор
3. int size() const; // размер
4. Перегрузить операторы
    a. Обеспечить доступ к элементам через [ ] (константный и обычный)
    b. operator = (const Vector&)
    c. operator+= (const Vector&)
    d. operator*= (const double d)

e. operator-= (const Vector&) реализовать через сложение с вектором умноженным на -1

5. Внешние операторы:
   a. double operator* (const Vector&, const Vector&)
   b. Vector operator* (double, const Vector&)
   c. Vector operator* (const Vector&, double)
   d. Vector operator+ (const Vector&, const Vector&)
   e. Vector operator- (const Vector&, const Vector&)
   f. Ввода-вывода в стандартные потоки. В случае ошибки ввода-вывода, исходный вектор переданный в качестве аргумента оператору >> должен оставаться неизменным. При этом оператор ввода вектора должен понимать формат { 5.0 , -3.0, 4.0 }

В случае операций над векторами разной размерности, меньший вектор расширять до размера большего добавляя нули.

Написать тестовую программу, демонстрирующую корректную работу векторов.

***Самостоятельная работа* № 5**

Задание 1. Требуется реализовать приложение Workflow Executor.

Workflow – вычислительная схема, состоящая из предопределенного набора вычислительных блоков и связей между ними. Программе подается workflow, описанный в файле.

В данной задаче рассматривается единственный тип workflow – линейный, т.е. конвейер.

Список блоков, используемых в схеме:

1. `readfile <filename>` – считывание текстового файла в память, целиком.
   Вход – отсутствует, выход – текст.
2. `writefile <filename>` – запись текста в файл.
   Вход – текст, выход – отсуствует.
3. `grep <word>` – выбор из входного текста строк, разделенных символами переноса строки, содержащих заданное слово `<word>`.
   Вход – текст, выход – текст.
4. `sort` – лексикогорафическая сортировка входного набора строк.
   Вход – текст, выход – текст.
5. `replace <word1> <word2>` – замена слова `word1` словом `word2` во входном тексте.
   Вход – текст, выход – текст.

Формат входного файла:

```
desc # описание блоков схемы
id1 = block1
id2 = block2
…
idN = blockN
csed
idA -> idB -> idC -> … -> idZ # описание структуры схемы
```

Где:

7. `desc, csed` – ключевые слова, ограничивающие раздел описания блоков workflow.
8. `id1 … idN` – целые, неотрицательные, неповторяющиеся числа.

9. `block1 … blockN –` блоки из списка блоков, с обязательными параметрами.
10. `idA, idB … idZ –` числа, принадлежащие множеству `id1…idN.` Могут повторяться, длина конвейера – неограничена.
11. `-> –` ключевое слово, обозначающее связь вычислительных узлов.

Ограничения на вычислительную схему:

- id1…idN – не повторяются.
- Блоки должны содержать нужное количество параметров.
- В описании структуры схемы первый блок должен быть блоком чтения из файла, последний – блоком записи в файл.
- Узлы схемы должны корректно соединяться, то есть тип данных на входе узла должен совпадать с типом данных предыдущего узла. Это означает, что чтение/запись файлов не может быть в середине схемы.

Пример:

workflow.txt

```
desc
1 = replace abracadabra cadabraabra
2 = grep braab
3 = sort
0 = readfile in.txt
5 = writefile out.txt
csed
0 -> 1 -> 2 -> 3 -> 5
```

Запуск:

```
executor.exe workflow.txt
```

Методические указания по реализации задачи:

1. Создать интерфейс Worker. Классы представляющие блоки схемы должны его имплементировать.
2. Для парсера схем и вычислителя/валидатора также выделить отдельные интерфейсы.
3. Для обработки ошибок и исключительных ситуаций использовать механизм исключений C++. Ознакомиться с классами исключений, определенных в STL.
4. Для работы с файлами использовать файловые потоки из STL.
5. При работе с потоками ввода-вывода, за состоянием потока наблюдать с помощью исключений. См. метод std::ios::exceptions (http://cplusplus.com/reference/iostream/ios/exceptions).

При сдаче задания продемонстрировать пять рабочих и пять нерабочих workflow.

6.2. Пример задания для реферативной самостоятельных работ:

1. Написать реферат на тему «Стиль программирования, использующий ассоциативную модель вычислений».
2. Написать реферат на тему «Средства поддержки программирования в сентенциальном стиле»

6.3. Примеры задания для реферативной самостоятельных работ:

Задание 1.

Вариант 1:

1. Какие элементы имеет машина фон Неймана?
2. Уровни доступа к памяти (перечислить)
3. Какие приведения вы знаете?
4. Что означает зависимость операторов?
5. Для чего используются подпрограммы?

Вариант 2:

1. Что означает однородность памяти?
2. Для чего служит кэш?
3. Что означает активность памяти?
4. Перечислите структуры управления традиционных языков
5. Какие виды модульности вы знаете?

Задание 2.

Вариант 1:

1. Охарактеризуйте отличия повелительного и изъявительного наклонения в языках программирования
2. Вычисления на основе систем продукций это …
3. Статическая и динамическая коммутация. В чем отличия?
4. Основа ассоциативной системы вычислений это …
5. В чем состоит различие конструктивного и классического доказательства теоремы?

Вариант 2:

1. Укажите побудительные причины нарушения канонической модели
2. За счет чего можно внедрять в языки программирования повелительное наклонение?
3. Как задаются вычисления на основе систем функций
4. Какие элементы коммутационной системы вычислений вы знаете
5. Элементарный акт аксиоматической системы вычислений это …

6.4. Вопросы к итоговой контрольной работе

1. Модель вычислений фон Неймана
2. Традиционные языки, наследующие модель вычислений фон Неймана
3. Модернизация канонической модели вычислений фон Неймана
4. Нетрадиционные модели вычислений: повелительное и изъявительное наклонение
5. Системы продукций. Системы функций. Коммутационные системы. Ассоциативные системы. Аксиоматические системы.
6. Ленивые вычисления и понятие необходимости вычислений.
7. Метод обобщения специфичного в функциональном программировании
8. Постулаты необходимости, их следствия. Конкретизации необходимости.
9. Решение численных задач в функциональном стиле: метод Ньютона-Рафсона: вычисление квадратного корня и численное дифференцирование
10. Ленивые и жадные вычисления в императивных языках
11. Ленивые вычисления: императивные примеры.
12. Определение сентенциального стиля программирования и его средств. Сопоставление с образцом.
13. Определение сентенциального стиля программирования и его средств. Абстрактный и конкретный синтаксис.
14. Логический вывод утверждений на основе фактов и язык Prolog
15. Фразовые формы, резолюция, унификация в языке Prolog

16. Семантика языка Prolog
17. Базы знаний в языке Prolog
18. Алгорифм Маркова и язык Рефал
19. Определение проецирования в Рефале
20. Применение продукции. Разрешение неоднозначностей
21. Дополнение основного механизма: детерминатив, внешние вычисления
22. Представление строк в Рефале
23. Задача и методы декомпозиции программ. Декомпозиция и моделирование. Виды декомпозиции
24. Концепция «Model View Controller»: определение и область применимости

7. Учебно-методическое и информационное обеспечение дисциплины

Основная литература:

1. Адаменко А., Кучуков А. *Логическое программирование и Visual Prolog (с CD).* — СПб.: БХВ-Петербург, 2003. — С. 990. —ISBN 5-94157-156-9
2. Братко И. *Алгоритмы искусственного интеллекта на языке PROLOG = Prolog Programming For Artificial Intelligence.* — М.: Вильямс, 2004. — 640 с. — ISBN 0-201-40375-7
3. Буч Г. и др. *Объектно-ориентированный анализ и проектирование с примерами приложений*. 3-е издание. — Бином, Невский диалект, Вильямс, 2010, ISBN 978-5-8459-1401-9, 0-201-89551-X, 0-8053-5340-2, 5-7989-0067-3, 5-7940-0017-1
4. Мейерс С. *Эффективное использование C++. 50 рекомендаций по улучшению ваших программ и проектов*. — ДМК, Питер, 2006, ISBN 0-201-92488-9, 5-93700-006-4, 5-469-01213-1
5. Мейерс С. *Эффективное использование C++. 35 новых способов улучшить стиль программирования*. — ДМК, Питер, .2006, ISBN 5-469-01215-8, 0-201-63371-X
6. Непейвода Н.Н. Скопин И.Н. *Основания программирования*. — Москва-Ижевск: Институт компьютерных исследований, 2003. — 868 с.
7. Страуструп Б. *Дизайн и эволюция языка C++*. — ДМК-Пресс, Питер, 2006, ISBN 5-469-01217-4, 0-201-54330-3
8. Страуструп Б. *Программирование принципы и практика использования C++*. — "И.Д. Вильямс", 2011, ISBN 978-5-8459-1621-1 (рус.)
9. Страуструп Б. *Язык программирования C++*. — Бином, Невский Диалект, 2008, ISBN 5-7989-0226-2, 5-7940-0064-3, 0-201-70073-5
10. J. Hughes. *Why Functional Programming Matters*. — The Computer Journal 1989, 32(2). — p. 98 – 107.

8. Материально-техническое обеспечение дисциплины

- Ноутбук, медиа-проектор, экран.
- Программное обеспечение для демонстрации слайд-презентаций.
- Компьютерные классы с предустановленным программным обеспечением для поддержки разработки C++ программ (например, MS Visual Studio)

Программа составлена в соответствии с требованиями ФГОС ВПО с учетом рекомендаций и ПрООП ВПО по направлению «**Ошибка! Источник ссылки не найден.**» и профилю подготовки «**Ошибка! Источник ссылки не найден.**».

Автор: _____Скопин Игорь Николаевич
к.ф.-м.н., доцент ММФ НГУ
с.н.с. ИВМиМГ СО РАН

Рецензент (ы) _____
_____

Программа одобрена на заседании _____
(Наименование уполномоченного органа вуза (УМК, НМС, Ученый совет)
от _____ года, протокол № _____