МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г. Шухова)

УТВЕРЖДАЮ
Директор института энергетики, информационных технологий и управляющих систем

канд. техн. наук, доцент _____ А.В. Белоусов

«11» _____ 2016 г.

РАБОЧАЯ ПРОГРАММА

дисциплины

ПРОГРАММИРОВАНИЕ И ОСНОВЫ АЛГОРИТМИЗАЦИИ

направление подготовки

13.03.02 Электроэнергетика и электротехника

Квалификация

бакалавр

Форма обучения

очная

Институт энергетики, информационных технологий и управляющих систем Кафедра электроэнергетики и автоматики

Белгород – 2016

Рабочая программа составлена на основании требований:

- Федерального государственного образовательного стандарта высшего образования по направлению подготовки 13.03.02 «Электроэнергетика и электротехника» (уровень бакалавриата), утвержденного приказом Министерства образования и науки Российской Федерации № 955 от 3 сентября 2015 г;
- плана учебного процесса БГТУ им. В.Г. Шухова, введенного в действие в 2016 году.

Составитель: канд. техн. наук _____ А.С. Солдатенков

Рабочая программа согласована с выпускающей кафедрой электроэнергетики и автоматики

Заведующий кафедрой: канд. техн. наук, доцент _____ А.В. Белоусов

«11» июня 2016 г.

Рабочая программа обсуждена на заседании кафедры электроэнергетики и автоматики

«11» июня 2016 г., протокол № 15

Заведующий кафедрой: канд. техн. наук, доцент _____ А.В. Белоусов

Рабочая программа одобрена методической комиссией института энергетики, информационных технологий и управляющих систем

«16» июня 2016 г., протокол № 2/16

Председатель: канд. техн. наук, доцент _____ А.Н. Семернин

1. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Формируемые компетенции			Требования к результатам обучения
№	Код компетенции	Компетенция	
Общепрофессиональные			
1	ОПК-2	Способность применять соответствующий физико-математический аппарат, методы анализа и моделирования, теоретического и экспериментального исследования при решении профессиональных задач	В результате освоения дисциплины обучающийся должен: **знать:** основные виды алгоритмических конструкций и способы их формализованного представления; структуру программы, алфавит, типы данных, синтаксис и семантику языка программирования С++, способы ввода и вывода информации; основы работы с динамически распределяемой памятью; алгоритмы сортировки и поиска данных; методы решения алгебраических и трансцендентных уравнений и их систем; принципы объектно-ориентированного программирования и подходы к построению иерархии классов, основанных на абстрактных моделях данных в С++; **уметь:** работать в инструментальной среде Microsoft Visual Studio 2015; объявлять и инициализировать переменные и константы, осуществлять ввод и вывод информации различного типа, применять базовые алгоритмические конструкции и операторы, создавать и использовать функции в программах на языке С++; выполнять сортировку и поиск данных в массивах, выбирать наилучшие алгоритмы на основе их содержимого; решать системы линейных алгебраических уравнений, вычислять определитель и обратную матрицу, находить норму вектора и матрицы; решать нелинейные уравнения и системы уравнений; объявлять классы, создавать и перегружать конструкторы, функции и операции в классах, а также выполнять одиночное наследование с применением виртуальных функций в С++; **владеть:** навыками создания, тестирования и отладки программ обработки данных, сортировки и поиска данных в массиве в соответствии с заданным критерием, решения систем линейных алгебраических уравнений, вычисления определителя и обратной матрицы, решения нелинейных уравнений и их систем на языке программирования С++ в инструментальной среде Microsoft Visual Studio 2015; навыками реализации на языке С++ пользовательских типов данных с применением объектно-ориентированного подхода при расчете и анализе электрических цепей.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Содержание дисциплины основывается и является логическим продолжением следующих дисциплин:

№	Наименование дисциплины (модуля)
1	Высшая математика
2	Информатика
3	Особенности профессиональной деятельности
4	Теоретические основы электротехники

Содержание дисциплины служит основой для изучения следующих дисциплин:

№	Наименование дисциплины (модуля)
1	Автоматизированные системы контроля и учета энергии
2	Управление электромеханическими системами
3	Умные энергетические микросети зданий
4	Релейная защита и автоматика
5	Коммутационные и защитные аппараты в системах электроснабжения

3. ОБЪЕМ ДИСЦИПЛИНЫ

Общая трудоемкость дисциплины составляет 5 зач. единиц, 180 часов.

Вид учебной работы	Всего часов	Семестр № 4
Общая трудоемкость дисциплины, час	180	180
Контактная работа (аудиторные занятия), в т.ч.:	68	68
лекции	34	34
лабораторные	34	34
практические	-	-
Самостоятельная работа студентов, в том числе:	112	112
курсовой проект	-	-
курсовая работа	-	-
расчетно-графическое задания	18	18
индивидуальное домашнее задание	-	-
другие виды самостоятельной работы	58	58
Форма промежуточная аттестация (зачет, экзамен)	36	экзамен (36)

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Наименование тем, их содержание и объем

Курс 2 Семестр 4

№ п/п	Наименование раздела (краткое содержание)	Объем на тематический раздел по видам учебной нагрузки, час			
		Лекции	Практические занятия	Лабораторные занятия	Самостоятельная работа
1.	Общие сведения о программировании на C++				
1.1	Понятие алгоритма, его свойства и виды. Базовые алгоритмические конструкции. Представление алгоритма в виде блок-схем и структурограмм. Основные этапы разработки программы. Языки программирования. История создания и развития языка C++. Стандарты на C++. Знакомство с IDE Microsoft Visual Studio 2015. Структура программы. Алфавит языка C++. Переменные и константы. Основные типы данных. Константы и литералы. Венгерская нотация. Макрокоманды *include* и *define*. Ввод и вывод данных. Форматирование вывода. Манипуляторы потоков. Особенности машинного представления чисел различных типов. Тестирование и отладка программ.	2		2	3
2.	Выражения и операторы				
2.1	Построение выражений в C++. Унарные и бинарные операции: арифметические, логические, сравнения, побитовые и др. Префиксная и постфиксная формы инкремента и декремента. Тернарная условная операция. Приоритет операций. Перечисляемые типы данных. Явное и неявное преобразование типов.	2		1	2
2.2	Понятие составного оператора. Условный оператор. Оператор множественного выбора. Использование оператора *break* в переключателях. Применение операторов ветвления в прикладном программировании. Вложенные условия.	2		4	5
2.3	Циклы с параметром *for*. Циклы с предусловием *while*. Циклы с постусловием *do*. Безусловные циклы. Использование операторов *break* и *continue* в циклах. Вложенные циклы. Применение циклов для итерационных вычислений.	2		3	4
3.	Функции				
3.1	Объявление функций. Прототипы функций. Вызов функций и возврат значений. Отличия от макросов. Передача	2		4	5

№ п/п	Наименование раздела (краткое содержание)	Объем на тематический раздел по видам учебной нагрузки, час			
		Лекции	Практические занятия	Лабораторные занятия	Самостоятельная работа
	параметров в функции. Формальные и фактические параметры. Глобальные и локальные переменные. Понятие области видимости и времени жизни переменной. Передача параметров по значению и по ссылке. Определение аргументов функции по умолчанию. Перегрузка функций. Встраивание функций как эффективный метод оптимизации программ. Рекурсия. Применение процедурного программирования в задачах энергетики и электротехники.				
4. Массивы и указатели					
4.1	Одномерные массивы. Объявление и инициализация. Доступ к элементам массива. Передача массивов в функции. Работа с массивами данных: удаление, вставка и перестановка элементов массива. Применение массивов для хранения и обработки различной информации.	2		4	5
4.2	Двумерные массивы. Объявление и инициализация. Работа с массивами данных: удаление, вставка и перестановка элементов массива. Многомерные массивы. Передача массивов в функции. Применение массивов для представления графов электрических схем.	2		4	5
4.3	Понятие указателя. Размещение и удаление указателя. Ссылки. Связь между массивами и указателями. Размещение и освобождение динамических массивов данных. Многомерные динамические массивы. Динамические строки с завершающим нулем. Применение динамических массивов для реализации промышленных протоколов взаимодействия с типовым периферийным оборудованием в составе супервизорных систем управления распределенными энергетическими системами.	2			3
4.4	Структуры и поля данных. Указатель на структуру. Создание динамических структур данных. Списки. Применение динамических структур данных в прикладном программировании.	2			2
5. Потоки и файлы					
5.1	Файловый ввод/вывод с использованием потоков. Создание, удаление и модификация файлов. Открытие и закрытие файла. Функции для обмена данными с файлами. Чтение и запись данных в файл. Ввод/вывод массивов данных.	2			1

№ п/п	Наименование раздела (краткое содержание)	Объем на тематический раздел по видам учебной нагрузки, час			
		Лекции	Практические занятия	Лабораторные занятия	Самостоятельная работа
	Применение потоков для представления информации в требуемом формате.				
6.	Основы алгоритмизации				
6.1	Сортировка методом простого выбора, простого обмена, методом прямых вставок, методом слияния, сортировка Хоара. Алгоритмы поиска данных в упорядоченных и неупорядоченных массивах: линейный поиск, барьерный поиск, метод половинного деления. Обход графов: поиск в ширину и глубину. Применение алгоритмов поиска для анализа топологии электрических цепей.	2		4	5
6.2	Математическое моделирование и вычислительный эксперимент. Численные методы как раздел современной математики. Роль компьютерно-ориентированных численных методов в исследовании сложных математических моделей, моделировании и анализе электрических цепей. Постановка задачи. Точные методы решения систем линейных алгебраических уравнений: метод Гаусса, метод Гаусса с выбором главного элемента, метод Гаусса-Жордана. Вычисление определителя матрицы. Вычисление обратной матрицы. Понятие нормы вектора и матрицы. Итерационные методы решения систем линейных алгебраических уравнений: метод Якоби (простых итераций), метод Зейделя. Сходимость итерационных методов.	2		3	5
6.3	Решение алгебраических и трансцендентных уравнений. Постановка задачи. Этапы отделения и уточнения корней. Метод половинного деления. Метод Ньютона. Метод хорд. Комбинированный метод. Метод простых итераций.	2		5	6
6.4	Решение систем нелинейных уравнений. Постановка задачи. Метод Ньютона. Метод простых итераций. Метод Зейделя. Сходимость итерационных методов. Применение численных методов в задачах энергетики.	2			1
7.	Основы объектно-ориентированного программирования				
7.1	Понятие классов, объектов, полей, методов и свойств. Абстракция данных. Наследование, инкапсуляция и полиморфизм. Доступ к членам класса. Конструкторы и деструкторы. Указатели на объекты. Использование классов.	2			2

№ п/п	Наименование раздела (краткое содержание)	Объем на тематический раздел по видам учебной нагрузки, час			
		Лекции	Практические занятия	Лабораторные занятия	Самостоятельная работа
7.2	Объявление и определение класса. Инициализация класса. Виды конструкторов. Перегрузка конструкторов. Передача экземпляров класса в функции. Неявный указатель *this*. Преобразования, определяемые классом. Создание пользовательских типов данных. Дружественные функции. Перегрузка бинарных и унарных операций. Статические члены данных.	2			2
7.3	Построение иерархии наследования. Доступ к базовым классам. Раннее и позднее связывание. Виртуальные функции. Абстрактные классы и чисто виртуальные функции. Подходы к построению иерархии классов при моделировании электрических цепей и проектировании автоматизированных систем управления и контроля распределенных энергетических объектов.	2			2
	ВСЕГО	34		34	58

4.2. Содержание практических (семинарских) занятий

Практические (семинарские) занятия учебным планом не предусмотрены.

4.3. Содержание лабораторных занятий

№ п/п	Наименование раздела дисциплины	Тема лабораторного занятия	К-во часов	К-во часов СРС
семестр № 4				
1	Общие сведения о программировании на С++. Выражения и операторы	Знакомство с инструментальной средой программирования Microsoft Visual Studio. Расчет схемы электрической цепи	4	4
2	Выражения и операторы	Базовые конструкции структурного программирования	6	6
3	Массивы и указатели	Одномерные массивы	4	4
4	Массивы и указатели	Двумерные массивы	4	4
5	Основы алгоритмизации	Сортировка и поиск данных	4	4
6	Функции. Основы алгоритмизации	Решение нелинейных алгебраических уравнений	6	6
7	Функции. Основы алгоритмизации	Решение систем линейных алгебраических уравнений	6	6
		ИТОГО	34	34

5. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

5.1. Перечень контрольных вопросов (типовых заданий)

№ п/п	Наименование раздела дисциплины	Содержание вопросов (типовых заданий)
1.	Общие сведения о программировании на С++	1. История развития языка программирования С++. Стандартизация языка С++. 2. Понятие переменной в С++. Основные типы данных. 3. Алфавит языка, литералы, константы и зарезервированные слова в С++. 4. Структура программы на языке С++. Применение комментариев в программе. Идентификаторы и Венгерская нотация. 5. Ввод/вывод информации с применением стандартных потоков. 6. Форматный ввод и вывод информации. Назначение флагов строки формата ввода/вывода. 7. Машинное представление чисел различных типов в ЭВМ.
2.	Выражения и операторы	8. Понятие выражения в С++. Операции, операнды и операторы. Постфиксная и префиксная формы записи операций. 9. Арифметические операции и побитовые операции. 10. Операции инкремента и декремента в префиксной и постфиксной формах записи. 11. Логические операции. Тернарный условный оператор. 12. Приоритет операций в С++. Изменение последовательности вычисления выражения. 13. Перечисляемый тип данных. Пример использования. 14. Явное и неявное приведение типов в С++. 15. Условный оператор. Полная и сокращенная запись. 16. Оператор множественного выбора. Пример использования. 17. Вложенные условия. Замена оператора множественного выбора вложенными условными операторами. 18. Представление фрагментов алгоритмов с ветвлениями и вложенными операторами множественного выбора в виде блок-схем. 19. Цикл с предусловием. Синтаксис и пример использования. 20. Цикл с постусловием. Синтаксис и пример использования. 21. Цикл с параметром for. Синтаксис и пример использования. 22. Вложенные циклы. Пример использования. 23. Применение операторов break и continue в циклах.
3.	Массивы и указатели	24. Объявление и инициализация одномерного массива в С++. 25. Работа с одномерными массивами Ввод и вывод значений

№ п/п	Наименование раздела дисциплины	Содержание вопросов (типовых заданий)
		элементов массивов. 26. Вставка в одномерный массив заданного элемента. 27. Удаление из одномерного массива заданного элемента. 28. Поиск в одномерном массиве заданного элемента. 29. Объявление и инициализация двумерного массива в C++. 30. Работа с двумерными массивами. Ввод и вывод значений элементов массива. 31. Вставка в двумерный массив заданной строки. 32. Удаление из двумерного массива заданного столбца. 33. Поиск в двумерном массиве заданного элемента. 34. Понятие указателя в C++. Размещение и удаление указателя. Ссылки. 35. Связь между массивами и указателями. Вычисление адреса элемента массива. 36. Размещение и освобождение одномерных динамических массивов в памяти. 37. Многомерные динамические массивы. Размещение и освобождение в памяти. 38. Динамические строки с завершающим нулем. 39. Стандартные функции C++ для работы со строками с завершающим нулем.
4.	Функции в C++	40. Понятие функции в C++. Объявление и определение функции. Вызов функции и возврат значений. 41. Понятие локальных и глобальных переменных. Область видимости и время жизни. 42. Передача параметров в функцию и возврат значений из функции. 43. Функции с параметрами по умолчанию. Ссылочные параметры. 44. Перегрузка функций. Пример использования. 45. Встраивание функций как эффективный метод оптимизации программ. 46. Макросы в C++ и их отличие от функций. 47. Понятие рекурсии. Примеры рекурсивных функций.
5.	Численные методы решения уравнений и их систем	48. Понятие математического моделирования и вычислительного эксперимента. 49. Точные методы решения систем линейных алгебраических уравнений: метод Гаусса. 50. Точные методы решения систем линейных алгебраических уравнений: метод Гаусса с выбором главного элемента. 51. Точные методы решения систем линейных алгебраических уравнений: метод Гаусса-Жордана. 52. Вычисление определителя матрицы методом Гаусса. 53. Вычисление обратной матрицы методом Гаусса.

№ п/п	Наименование раздела дисциплины	Содержание вопросов (типовых заданий)
		54. Понятие нормы вектора и матрицы. Способы вычисления. 55. Итерационные методы решения систем линейных алгебраических уравнений: метод Якоби (простых итераций). 56. Итерационные методы решения систем линейных алгебраических уравнений: метод Зейделя. 57. Решение алгебраических уравнений. Постановка задачи. Этапы отделения и уточнения корней. 58. Решение нелинейных уравнений. Метод половинного деления. 59. Решение нелинейных уравнений. Метод Ньютона. 60. Решение нелинейных уравнений. Метод хорд. 61. Решение нелинейных уравнений. Комбинированный метод. 62. Решение нелинейных уравнений. Метод простых итераций. 63. Решение систем нелинейных уравнений. Метод Ньютона.
6.	Потоки и файлы	64. Файловая организация программ на C++. Понятие заголовочного файла. Макрокоманда include. 65. Создание и использования пользовательской библиотеки функций для определения частотных характеристик электрической цепи. 66. Файловый ввод с использованием потоков в C++. Пример использования. 67. Файловый вывод с использованием потоков в C++. Пример использования. 68. Создание и удаление файла в C++. 69. Открытие и закрытие файла. Работа с файлом. Режимы. 70. Манипуляторы потоков в C++. Назначение и примеры использования. 71. Работа с текстовыми файлами в C++. Отличия текстового файла от бинарного.
7.	Алгоритмы сортировки и поиска. Динамические структуры данных	72. Сортировка массива методом простого выбора. 73. Сортировка массива методом простого обмена. 74. Сортировка массива методом прямых вставок. 75. Сортировка массива методом слияния. 76. Сортировка массива методом Хоара. 77. Линейный и барьерный поиск данных в массиве. 78. Поиск в массиве методом половинного деления. 79. Структура в C++. Назначение и отличия от массивов. Размещение и освобождение в памяти. 80. Однонаправленные списки. Пример создания использования. 81. Двунаправленные списки. Пример создания использования.
8.	Основы объектно-ориентированного программирования	82. Понятие наследования, инкапсуляции и полиморфизма в объектно-ориентированном программировании. 83. Создание класса в C++. Методы и свойства класса. 84. Открытые и закрытые элементы класса. Уровни доступа.

№ п/п	Наименование раздела дисциплины	Содержание вопросов (типовых заданий)
		85. Конструкторы и деструкторы. Назначение и применение. 86. Виды конструкторов. Отличия и примеры применения. 87. Объявление и определение экземпляра класса в C++. 88. Перегрузка конструкторов классов. Пример использования. 89. Перегрузка унарных операций. Пример использования. 90. Перегрузка бинарных операций. Пример использования. 91. Применение дружественных функций в C++. 92. Применение механизма наследования при построении иерархии классов в C++. Доступ к базовым классам. 93. Виртуальные функции классов в C++. Пример применения.

5.2. Перечень тем курсовых проектов, курсовых работ, их краткое содержание и объем

Курсовые проекты (работы) учебным планом не предусмотрены.

5.3. Перечень индивидуальных домашних заданий, расчетно-графических заданий

Учебным планом предусмотрено одно расчетно-графическое задание, целью которого является привитие навыков разработки программного обеспечения по расчету разветвленных электрических цепей, образованных совокупностью обобщенных ветвей с несколькими источниками синусоидальных ЭДС и тока в установившемся режиме. Подобное программное обеспечение может применяться в задачах расчета и моделирования элементов и систем в электроэнергетике и электротехнике, а также как составная часть программного обеспечения автоматизированных систем управления и контроля (мониторинга) распределенных энергосистем.

Для заданной электрической цепи, параметры которой представлены в таблице, необходимо, используя метод узловых напряжений с применением топологических матриц, написать программу расчета токов во всех ветвях схемы и выполнить проверку правильности расчета с помощью баланса мощностей. Программа должна иметь объектно-ориентированную структуру и обеспечивать функциональные возможности ввода исходных данных из файла и вывода результатов расчета на экран и в текстовый файл.

Отчет должен содержать:

- чертеж схемы электрической цепи с обозначением узлов и токов в ветвях;
- результаты ручного расчета схемы (топологические матрицы, узловые потенциалы, токи в ветвях, баланс мощностей);
- листинг программы со всеми используемыми модулями;

- снимок экрана с результатами работы программы, содержащий значения узловых потенциалов, токов в ветвях, и данные о балансе мощностей;
- описание используемых подпрограмм, входных и выходных данных, возвращаемых значений функций;
- блок-схемы алгоритмов работы программы и всех подпрограмм.

Структура входного и выходного файлов может быть выбрана произвольно. Решение системы уравнений рекомендуется выполнять с помощью любого итерационного метода, однако допускается применение точных методов (в том числе путем нахождения обратной матрицы).

Для расчетных токов в каждой ветви необходимо привести комплексные и действующие значения. Во всех случаях считать, что взаимоиндукцией между ветвями электрической цепи можно пренебречь, а все элементы схемы идеальные.

Пример типового задания

Ветвь и направление тока в ней	Параметры нагрузки			Параметры источников ЭДС и тока				
	R, Ом	L, мГн	C, мкФ	E, В	φ_E, °	J, А	φ_J, °	f, Гц
1→2	100	12	11			0.2	10	30
1→3	12			80	16			30
1→4	22	22	15					
2→4	15	56	39					
2→5	82	56						
3→4	51		56					
3→6	39			40	-22			30
4→5	47	33	11					
4→6	20	12	16					
4→7	10	47	91					
5→7	18			-60	30			30
6→7	36			40	75			30

Схема электрической цепи, содержащая 7 узлов и 6 независимых контуров, представлена в виде таблицы (по вариантам), каждая строка которой описывает параметры соответствующей ветви. В столбце 1 указаны заданные направления токов в ветвях, соединяющих соответствующие узлы. Параметры нагрузочных сопротивлений в ветвях схемы представлены в столбцах 2-4, а параметры идеальных источников ЭДС и тока – в столбцах 5-9 (действующие значения и начальные фазы в градусах, в столбце 9 – частота). Все источники ЭДС включены последовательно с нагрузкой и положительное направление ЭДС совпадает с соответствующим направлением, указанным в столбце 1. Источники тока (если они есть) включены параллельно нагрузке и направление тока соответствующего источника противоположно направлению, указанному в столбце 1. Во всех случаях знак минус перед действующим значением ЭДС или тока в столбцах 5 и 7 означает включение соответствующего источника в противоположном направлении относительно заданного в столбце 1.

5.4. Перечень контрольных работ

Контрольные работы учебным планом не предусмотрены.

6. ОСНОВНАЯ И ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

6.1. Перечень основной литературы

1. Программирование на языке высокого уровня C/C++ [Электронный ресурс]: конспект лекций/. – Электрон. текстовые данные. – М.: Московский государственный строительный университет, Ай Пи Эр Медиа, ЭБС АСВ, 2016. – 140 с. – 978-5-7264-1285-6. – Режим доступа: http://www.iprbookshop.ru/48037.html

2. Сундукова Т.О. Структуры и алгоритмы компьютерной обработки данных [Электронный ресурс]/ Т.О. Сундукова, Г.В. Ваныкина. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 749 с. – 2227-8397. – Режим доступа: http://www.iprbookshop.ru/57384.html

3. Шелупанов А.А. Информатика. Базовый курс. Часть 3. Основы алгоритмизации и программирования в среде Visual C++ 2005 [Электронный ресурс]: учебник/ А.А. Шелупанов, В.Н. Кирнос. – Электрон. текстовые данные. – Томск: Томский государственный университет систем управления и радиоэлектроники, В-Спектр, 2008. – 216 с. – 978-5-91191-091-4. – Режим доступа: http://www.iprbookshop.ru/14013.html

4. Зенков А.В. Численные методы [Электронный ресурс] : учебное пособие / А.В. Зенков. — Электрон. текстовые данные. — Екатеринбург: Уральский федеральный университет, 2016. — 124 с. — 978-5-7996-1781-3. — Режим доступа: http://www.iprbookshop.ru/68315.html

6.2. Перечень дополнительной литературы

1. Шишкин А.Д. Программирование на языке Си [Электронный ресурс]: учебное пособие/ А.Д. Шишкин. – Электрон. текстовые данные. – СПб.: Российский государственный гидрометеорологический университет, 2003. – 104 с. – 2227-8397. – Режим доступа: http://www.iprbookshop.ru/17959.html

2. Барышникова М.Ю. Основы программирования на C/C++. Часть 2 [Электронный ресурс]: учебное пособие/ М.Ю. Барышникова, А.В. Силантьева. – Электрон. текстовые данные. – М.: Московский государственный технический университет имени Н.Э. Баумана, 2007. – 70 с. – 978-5-7038-2939-4. – Режим доступа: http://www.iprbookshop.ru/31492.html

3. Иванов В.Б. Прикладное программирование на C/C++. С нуля до мультимедийных и сетевых приложений [Электронный ресурс]/ В.Б. Иванов. – Электрон. текстовые данные. – М.: СОЛОН-ПРЕСС, 2011. – 240 с. – 5-98003-279-7. – Режим доступа: http://www.iprbookshop.ru/65139.html

4. Костомаров Д.П. Программирование и численные методы [Электронный

ресурс]: учебное пособие/ Д.П. Костомаров, Л.С Корухова, С.Г. Манжелей. – Электрон. текстовые данные. – М.: Московский государственный университет имени М.В. Ломоносова, 2001. – 224 с. – 5-211-04059-7. – Режим доступа: http://www.iprbookshop.ru/13108.html

5. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 с. – Режим доступа: http://www.iprbookshop.ru/39552

6. Батищев Р.В. Структуры и алгоритмы обработки данных. Часть 1 [Электронный ресурс]: учебное пособие/ Р.В. Батищев. – Электрон. текстовые данные. – Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2014. – 90 с. – 5-88247-716-6. – Режим доступа: http://www.iprbookshop.ru/55658.html

7. Комлев Н.Ю. Полезное программирование [Электронный ресурс]/ Н.Ю. Комлев. – Электрон. текстовые данные. – М.: СОЛОН-ПРЕСС, 2016. – 256 с. – 978-5-91359-171-5. – Режим доступа: http://www.iprbookshop.ru/53837.html

8. Токманцев Т.Б. Алгоритмические языки и программирование [Электронный ресурс]: учебное пособие/ Т.Б. Токманцев. – Электрон. текстовые данные. – Екатеринбург: Уральский федеральный университет, 2013. – 104 с. – 978-5-7996-1023-4. – Режим доступа: http://www.iprbookshop.ru/68220.html

9. Шевцов Г.С. Численные методы линейной алгебры. [Электронный ресурс] / Г.С. Шевцов, О.Г. Крюкова, Б.И. Мызникова. – Электрон. дан. – СПб.: Лань, 2011. – 496 с. – Режим доступа: http://e.lanbook.com/book/1800 – Загл. с экрана.

6.3. Перечень интернет ресурсов

1. Каталог образовательных Интернет-ресурсов: Программирование [Электронный ресурс]. – Режим доступа: http://window.edu.ru/catalog?p_rubr=2.2.75.6.14. – Заглавие с экрана.
2. ISO/IEC JTC1/SC22/WG21 – Комитет по стандартизации C++. ISO/IEC 14882:1998(E) Язык программирования C++ (англ.) [Электронный ресурс]. – Режим доступа: http://www.open-std.org/jtc1/sc22/wg21/. – Заглавие с экрана.
3. Стандартные библиотеки и язык C++ [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/hh875057.aspx. – Заглавие с экрана.
4. Портал о программировании Code-Live. C++ с нуля [Электронный ресурс]. – Режим доступа: https://code-live.ru/tag/cpp-manual/. – Заглавие с экрана.
5. C++ reference. C reference [Электронный ресурс]. – Режим доступа: http://en.cppreference.com/w/. – Заглавие с экрана.
6. Основы программирования на языках Си и C++ для начинающих [Электронный ресурс]. – Режим доступа: http://cppstudio.com. – Заглавие с экрана.
7. Руководства и справочные материалы по C/C++ [Электронный ресурс]. – Режим доступа: http://www.codenet.ru/cat/Languages/C-CPP/Tutorials/. – Заглавие с экрана.

8. Хабрахабр, крупнейший в Европе ресурс для IT-специалистов [Электронный ресурс]. – Режим доступа: https://habrahabr.ru. – Заглавие с экрана.

9. Online Documentation - Developer Express Inc [Электронный ресурс]. – Режим доступа: https://documentation.devexpress.com/. – Заглавие с экрана.

10. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: https://www.visualstudio.com/ru/. – Заглавие с экрана.

11. Вычислительные методы [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Вычислительные_методы. – Заглавие с экрана.

12. Фридман, А. Язык программирования C++: Информация [Электронный ресурс] / Фридман А. // Национальный Открытый Университет «ИНТУИТ». – Режим доступа: http://www.intuit.ru/studies/courses/17/17/info. – Заглавие с экрана.

13. АЛЁНА C++. Программирование для прагматиков [Электронный ресурс]. – Режим доступа: http://alenacpp.blogspot.ru. – Заглавие с экрана.

14. Объектно-ориентированное программирование (ООП) в C++ [Электронный ресурс]: Обучение программированию. Для студентов математического факультета МПГУ (и всех желающих обучаться по материалам данного сайта самостоятельно). – Режим доступа: http://itedu.ru/courses/cpp/oop-in-cpp. – Заглавие с экрана.

15. C++. Форум программистов C++. Обсуждение языка программирования C++. Помощь в решении задач, ответы на вопросы [Электронный ресурс]. – Режим доступа: http://www.cyberforum.ru/cpp/. – Заглавие с экрана.

16. Язык C++ [Электронный ресурс]. – Режим доступа: http://prog-cpp.ru/cpp/. – Заглавие с экрана.

17. Программирование C++ [Электронный ресурс]. – Режим доступа: http://function-x.ru/comp_prog_cpp.html. – Заглавие с экрана.

7. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

Лекционные занятия – поточная аудитория, оснащенная доской и презентационной техникой (ноутбук, проектор, экран), комплектом электронных презентаций.

Лабораторные занятия – специализированный компьютерный класс М229, оснащенный презентационной техникой и персональными компьютерами (Intel Core i7-3770/ H81/ 8192Mb/ 1Tb/ 21.5”IPS/ Wi-Fi/ LAN100Mb/DWD-RW), подключенными к локальной сети университета с доступом в интернет.

Для лекционных и лабораторных занятий используется предустановленное лицензионное программное обеспечение Microsoft: Windows 7 Professional (№ дог. 63-14к от 02.07.2014), Office 2013 Professional (№ дог. 31401445414 от 25.09.2014) и Visual Studio 2013 (№ дог. 63-14к от 02.07.2014).

Для самостоятельной работы студентов предусмотрен компьютерный класс, оснащенный компьютерной техникой с возможностью подключения к сети «Интернет», обеспечением доступа в электронную информационно-образовательную среду университета, а так же участием в программах Microsoft Office 365 для образования (студенческий) (№ дог. E04002C51M) с возможностью бесплатной загрузки программного обеспечения Microsoft.

лицензионное программное обеспечение Microsoft: Windows 10 Корпоративная (Enterprise) (№ дог. E04002C51M), Office Professional Plus 2016 (№ дог. E04002C51M), Visio Professional 2013 (№ дог. E04002C51M), Visual Studio 2015 (№ дог. E04002GR24), а также PTC MathCad Prime 4.0 Express (распространяется свободно), Matlab 2013b v.8.2.0.701 (№ дог. 362444), SMath Studio 0.98 (сборка 6484) (распространяется свободно).

Для самостоятельной работы студентов предусмотрен компьютерный класс, оснащенный компьютерной техникой с возможностью подключения к сети «Интернет», обеспечением доступа в электронную информационно-образовательную среду университета, а так же участием в программах Microsoft Office 365 для образования (студенческий) (№ дог. E04002C51M) с возможностью бесплатной загрузки программного обеспечения Microsoft.

8. УТВЕРЖДЕНИЕ РАБОЧЕЙ ПРОГРАММЫ

Рабочая программа с изменениями, дополнениями утверждена на 2017/2018 учебный год.

Протокол № 14 заседания кафедры от « 10 » 06 2017 г.

Заведующий кафедрой ЭиА _____ А.В. Белоусов

Директор института ЭИТУС _____ А.В. Белоусов

Список изменений и дополнений в рабочей программе.

В пункт 6.2 добавлены следующие литературные источники:

1. Костюкова Н.И. Программирование на языке Си [Электронный ресурс]: методические рекомендации и задачи по программированию/ Н.И. Костюкова. – Электрон. текстовые данные. – Новосибирск: Сибирское университетское издательство, 2017. – 160 с. – 978-5-379-02016-3. – Режим доступа: http://www.iprbookshop.ru/65289.html

В пункт 6.3 добавлены следующие интернет-источники:

1. Документация и книги по программированию [Электронный ресурс]. – Режим доступа: http://www.helloworld.ru/. – Заглавие с экрана.
2. Алгоритмы в C++ [Электронный ресурс]: PureCodeCpp. Основы программирования C++ для начинающих. – Режим доступа: http://purecodecpp.com/algoritmy-v-c. – Заглавие с экрана.
3. Руководство по языку программирования C++ [Электронный ресурс]: METANIT.COM Сайт о программировании. – Режим доступа: https://metanit.com/cpp/tutorial/. – Заглавие с экрана.

Пункт 7 заменен следующим содержанием:

Лекционные занятия – поточная аудитория, оснащенная доской и презентационной техникой (ноутбук, проектор, экран), комплектом электронных презентаций.

Лабораторные занятия – специализированный компьютерный класс М424, оснащенный презентационной техникой (проектор Acer Projector P1165) и персональными компьютерами (Intel Core i3-8100 CPU 3.60 ГГц/ Gigabyte Z370 HD3/ RAM 8192 Мб/ HDD 1 Тб/ NVIDIA GeForce GTX 750/ AOC 23,8"/ ASUS DRW-24D5MT/ Wi-Fi/ LAN100Mb/ CyberPower BS850E), подключенными к локальной сети университета с доступом в интернет.

Для лекционных и лабораторных занятий используется предустановленное

Рабочая программа с изменениями, дополнениями утверждена на 2018/2019 учебный год.

Протокол № 10 заседания кафедры от «14» 05 2018 г.

Заведующий кафедрой ЭиА _____ А.В. Белоусов

Директор института ЭИТУС _____ А.В. Белоусов

Список изменений и дополнений в рабочей программе.

В пункт 6.1 добавлены следующие литературные источники:

1. Зоткин С.П. Программирование на языке высокого уровня C/C++ [Электронный ресурс] : конспект лекций / С.П. Зоткин. — 3-е изд. — Электрон. текстовые данные. — М. : МИСИ-МГСУ, ЭБС АСВ, 2018. — 140 с. — 978-5-7264-1810-0. — Режим доступа: http://www.iprbookshop.ru/76390.html

В пункт 6.2 добавлены следующие литературные источники:

1. Белева Л.Ф. Программирование на языке C++ [Электронный ресурс] : учебное пособие / Л.Ф. Белева. — Электрон. текстовые данные. — Саратов: Ай Пи Эр Медиа, 2018. — 81 с. — 978-5-4486-0253-5. — Режим доступа: http://www.iprbookshop.ru/72466.html

Рабочая программа без изменений утверждена на 2019/2020 учебный год.

Протокол № 13 заседания кафедры от « 07 » июня 2019 г.

Заведующий кафедрой ЭиА _____ А.В. Белоусов

Директор института ЭИТУС _____ А.В. Белоусов

8. УТВЕРЖДЕНИЕ РАБОЧЕЙ ПРОГРАММЫ

Рабочая программа без изменений утверждена на 2020/2021 учебный год.

Протокол № 10 заседания кафедры от «14» мая 2020 г.

Заведующий кафедрой _____ А.В. Белоусов
подпись, ФИО

Директор института _____ А.В. Белоусов
подпись, ФИО

Утверждение рабочей программы без изменений.

Рабочая программа без изменений утверждена на 2021/2022 учебный год.

Протокол № 11 заседания кафедры от «15» мая 2021 г.

Заведующий кафедрой _____ А.В. Белоусов

Директор института _____ А.В. Белоусов

ПРИЛОЖЕНИЕ

Методические указания для обучающегося по освоению дисциплины.

Курс "Программирование и основы алгоритмизации" предполагает ознакомление студентов с основными видами, этапами проектирования и жизненного цикла программных продуктов; синтаксисом и семантикой алгоритмического языка программирования высокого уровня C++; структурным и модульным программированием; типизацией и структуризацией программных данных; статическими и динамическими данными; потоками ввода-вывода; файлами; проектированием программных алгоритмов (основными принципами и подходами); классами алгоритмов; методами частных целей, подъема ветвей и границ, эвристикой; рекурсией и итерацией; сортировкой и поиском данных; методами и средствами объектно-ориентированного программирования; стандартами на разработку прикладных программных средств; документированием, сопровождением и эксплуатацией программных средств.

Занятия проводятся в виде лекций и лабораторных занятий. Для закрепления практических навыков предусмотрен текущий и итоговый контроль. Текущий контроль знаний проводится в форме защиты лабораторных работ и выполнения расчетно-графической работы, а формой итогового контроля является экзамен. Важное значение для изучения курса имеет самостоятельная работа, в рамках которой рекомендуется не только познакомиться с обязательными и дополнительными источниками литературы, но также на практике опробовать материалы лекций.

Перед началом лекционных занятий студент должен самостоятельно ознакомиться с изучаемой темой, используя учебник или учебные пособия, включая рекомендованные электронные ресурсы. Во время лекции студент должен внимательно слушать преподавателя и конспектировать лекционный материал. В конце занятия, при необходимости, задать вопросы по изучаемой теме. Рекомендуется на 1 час лекции затрачивать не менее 0,5 часа самостоятельной работы. После лекции студент самостоятельно должен прочитать конспект лекции и сопоставить с материалом учебника или учебного пособия с целью полного усвоения изучаемой темы. При подготовке к лабораторным занятиям студентам необходимо самостоятельно подготовится к выполнению работ (написанию программ), используя методические указания и электронные раздаточные материалы. Рекомендуется на 1 час лабораторных занятий затрачивать не менее 1 часа самостоятельной работы.

Самостоятельное изучение материалов курса, включая язык программирования C++, рекомендуется проводить с использованием указанных Интернет-ресурсов, например Microsoft MSDN, cppstudio, codenet, habrahabr и др. Хорошим подспорьем может считаться участие в интернет-конференциях

(форумах) cyberforum и ixbt, где можно вынести на обсуждение интернет-сообщества те или иные аспекты программирования на языке C++.

Изученный в рамках данного курса материал, может быть полезен при выполнении выпускной квалификационной работы.

Ниже представлена выдержка ряда замечаний и рекомендаций от автора языка C++ (*Бьерн Страуструп. Язык программирования C++. Специальное издание. Пер. с англ. – М.: Издательство Бином, 2011 г. – 1136 с.: ил.*).

Замечания по реализации

Существует несколько распространяемых независимых реализаций C++. Появилось большое число сервисных программ, библиотек и интегрированных систем программирования. Имеется масса книг, руководств, журналов, статей, сообщений по электронной почте, технических бюллетеней, отчетов о конференциях и курсов, из которых можно получить все необходимые сведения о последних изменениях в C++, его использовании, сервисных программах, библиотеках, новых трансляторах и т.д. Однако в любом случае стоит получить доступ хотя бы к двум источникам информации, поскольку у каждого источника может быть своя позиция.

Замечания по проекту языка

При разработке языка C++ одним из важнейших критериев выбора была простота. Когда возникал вопрос, что упростить: руководство по языку и другую документацию или транслятор, - то выбор делали в пользу первого. Огромное значение придавалось совместимости с языком C, что помешало удалить его синтаксис. В C++ нет типов данных и элементарных операций высокого уровня. Например, не существует типа матрица с операцией обращения или типа строка с операцией конкатенации. Если пользователю понадобятся подобные типы, он может определить их в самом языке. Программирование на C++ по сути сводится к определению универсальных или зависящих от области приложения типов. Хорошо продуманный пользовательский тип отличается от встроенного типа только способом определения, но не способом применения. Из языка исключались возможности, которые могут привести к накладным расходам памяти или времени выполнения, даже если они непосредственно не используются в программе. Например, было отвергнуто предложение хранить в каждом объекте некоторую служебную информацию. Если пользователь описал структуру, содержащую две величины, занимающие по 16 разрядов, то гарантируется, что она поместится в 32-х разрядный регистр.

Историческая справка

Безусловно C++ многим обязан языку C, который сохраняется как его подмножество. Сохранены и все свойственные C средства низкого уровня, предназначенные для решения самых насущных задач системного программирования. Еще одним источником вдохновения был язык SIMULA-67,

именно из него была заимствована концепция классов (вместе с производными классами и виртуальными функциями). Возможность в C++ перегрузки операций и свобода размещения описаний всюду, где может встречаться оператор, напоминают язык Алгол-68. С момента выхода в свет язык C++ подвергся существенным изменениям и уточнениям. В основном это касается разрешения неоднозначности при перегрузке, связывании и управлении памятью. Вместе с тем, были внесены незначительные изменения с целью увеличить совместимость с языком С. Были также введены некоторые обобщения и существенные расширения, как то: множественное наследование, функции-члены со спецификациями static и const, защищенные члены (protected), шаблоны типа и обработка особых ситуаций. Все эти расширения и доработки были нацелены на то, чтобы C++ стал языком, на котором можно создавать и использовать библиотеки. Шаблоны типов появились частично из-за желания формализовать макросредства, а частично были инспирированы описанием генерических объектов в языке Ада (с учетом их достоинств и недостатков) и параметризированными модулями языка CLU. Механизм обработки особых ситуаций появился отчасти под влиянием языков Ада и CLU, а отчасти под влиянием ML. Другие расширения, введенные за период между 1985 и 1991 г.г. (такие как множественное наследование, статические функции-члены и чистые виртуальные функции), скорее появились в результате обобщения опыта программирования на C++, чем были почерпнуты из других языков.

Название C++ (си плюс плюс), было придумано Риком Маскитти летом 1983 г. Это название отражает эволюционный характер изменений языка С. Обозначение ++ относится к операции наращивания С. Чуть более короткое имя C+ является синтаксической ошибкой. Кроме того, оно уже было использовано как название совсем другого языка. Знатоки семантики С находят, что C++ хуже, чем ++C. Язык не получил названия D, поскольку он является расширением С, и в нем не делается попыток решить какие-либо проблемы за счет отказа от возможностей С. Изначально C++ был задуман для того, чтобы не надо было программировать на ассемблере, С или других современных языках высокого уровня. Основное его предназначение - упростить и сделать более приятным процесс программирования для отдельного программиста.

В связи с лавинообразным процессом увеличения числа пользователей C++, были сделаны следующие изменения. Примерно в 1987 г. стало очевидно, что работа по стандартизации C++ неизбежна и что следует незамедлительно приступить к созданию основы для нее. В результате были предприняты целенаправленные действия, чтобы установить контакт между разработчиками C++ и большинством пользователей. Фирма AT&T Bell Laboratories внесла основной вклад в эту работу, предоставив автору право изучать версии справочного руководства по языку вместе с упоминавшимися разработчиками и пользователями. Не следует недооценивать этот вклад, т.к. многие из них

работают в компаниях, которые можно считать конкурентами фирмы AT&T. Менее просвещенная компания могла бы просто ничего не делать, и в результате появилось бы несколько несогласованных версий языка. Около ста представителей из порядка 20 организаций изучали и комментировали то, что стало современной версией справочного руководства и исходными материалами для ANSI по стандартизации C++.

Сравнение языков C++ и C

Выбор C в качестве базового языка для C++ объясняется следующими его достоинствами: универсальность, краткость и относительно низкий уровень, адекватность большинству задач системного программирования, он идет в любой системе и на любой машине, полностью подходит для программной среды UNIX.

В C существуют свои проблемы, но в языке, разрабатываемом «с нуля» они появились бы тоже, а проблемы C, по крайней мере, хорошо известны. C++ стал использоваться шире, но по мере роста его возможностей, выходящих за пределы C, вновь и вновь возникала проблема совместимости. Ясно, что отказавшись от части наследства C, можно избежать некоторых проблем. Это не было сделано по следующим причинам: существуют миллионы строк программ на C, которые можно улучшить с помощью C++, но при условии, что полной переписи их на язык C++ не потребуется; существуют миллионы строк библиотечных функций и служебных программ на C, которые можно было бы использовать в C++ при условиях совместимости обоих языков на стадии связывания и их большого синтаксического сходства; существуют сотни тысяч программистов, знающих C; им достаточно овладеть только новыми средствами C++ и не надо изучать основ языка; поскольку C и C++ будут использоваться одними и теми же людьми на одних и тех же системах многие годы, различия между языками должны быть либо минимальными, либо максимальными, чтобы свести к минимуму количество ошибок и недоразумений.

Описание C++ было переработано так, чтобы гарантировать, что любая допустимая в обоих языках конструкция означала в них одно и то же. Язык C сам развивался в последние несколько лет, что отчасти было связано с разработкой C++. Стандарт ANSI для C содержит, например, синтаксис описания функций, позаимствованный из языка «C с классами». Происходит взаимное заимствование, например, тип указателя void* был придуман для ANSI C, а впервые реализован в C++. В идеале C++ должен максимально приближаться к ANSI C, но не более. Стопроцентной совместимости никогда не было и не будет, поскольку это нарушит надежность типов и согласованность использования встроенных и пользовательских типов, а эти свойства всегда были одними из главных для C++.

Для изучения C++ не обязательно знать C. Программирование на C способствует усвоению приемов и даже трюков, которые при программировании

на C++ становятся просто ненужными. Например, явное преобразование типа (приведение), в C++ нужно гораздо реже, чем в C. Тем не менее, хорошие программы на языке C по сути являются программами на C++. В процессе изучения C++ будет полезен опыт работы с любым языком со статическими типами.

Эффективность и структура

Развитие языка C++ происходило на базе языка C, и, за небольшим исключением, C был сохранен в качестве подмножества C++. Базовый язык C был спроектирован таким образом, что имеется очень тесная связь между типами, операциями, операторами и объектами, с которыми непосредственно работает машина, т.е. числами, символами и адресами. За исключением операций new, delete и throw, а также проверяемого блока, для выполнения операторов и выражений C++ не требуется скрытой динамической аппаратной или программной поддержки. В C++ используется та же (или даже более эффективная) последовательность команд для вызова функций и возврата из них, что и в C. Если даже эти довольно эффективные операции становятся слишком дорогими, то вызов функции может быть заменен подстановкой ее тела, причем сохраняется удобная функциональная запись безо всяких расходов на вызов функции.

Первоначально язык C задумывался как конкурент ассемблера, способный вытеснить его из основных и наиболее требовательных к ресурсам задач системного программирования. В проекте C++ были приняты меры, чтобы успехи C в этой области не оказались под угрозой. Различие между двумя языками прежде все состоит в степени внимания, уделяемого типам и структурам. Язык C выразителен и в то же время снисходителен по отношению к типам. Язык C++ еще более выразителен, но такой выразительности можно достичь лишь тогда, когда типам уделяют большое внимание. Когда типы объектов известны, транслятор правильно распознает такие выражения, в которых иначе программисту пришлось бы записывать операции с утомительными подробностями. Кроме того, знание типов позволяет транслятору обнаруживать такие ошибки, которые в противном случае были бы выявлены только при тестировании. Само по себе использование строгой типизации языка для контроля параметров функции, защиты данных от незаконного доступа, определения новых типов и операций не влечет дополнительных расходов памяти и увеличения времени выполнения программы.

В проекте C++ особое внимание уделяется структурированию программы. Это вызвано увеличением размеров программ со времени появления C. Небольшую программу (скажем, не более 1000 строк) можно заставить из упрямства работать, нарушая все правила хорошего стиля программирования. Однако, действуя так, человек уже не сможет справиться с большой программой.

Если у вашей программы в 10 000 строк плохая структура, то вы обнаружите, что новые ошибки появляются в ней так же быстро, как удаляются старые. C++ создавался с целью, чтобы большую программу можно было структурировать таким образом, чтобы одному человеку не пришлось работать с текстом в 25000 строк. В настоящее время можно считать, что эта цель полностью достигнута. Существуют, конечно, программы еще большего размера. Однако те из них, которые действительно используются, обычно можно разбить на несколько практически независимых частей, каждая из которых имеет значительно меньший упомянутого размер. Естественно, трудность написания и сопровождения программы определяется не только числом строк текста, но и сложностью предметной области.

К сожалению, не всякую часть программы можно хорошо структурировать, сделать независимой от аппаратуры, достаточно понятной и т.д. В C++ есть средства, непосредственно и эффективно представляющие аппаратные возможности. Их использование позволяет избавиться от беспокойства о надежности и простоте понимания программы. Такие части программы можно скрывать, предоставляя надежный и простой интерфейс с ними.

Философские замечания

Язык программирования решает две взаимосвязанные задачи: позволяет программисту записать подлежащие выполнению действия и формирует понятия, которыми программист оперирует, размышляя о своей задаче. Первой цели идеально отвечает язык, который очень «близок машине». Тогда со всеми ее основными «сущностями» можно просто и эффективно работать на этом языке, причем делая это очевидным для программиста способом. Именно это имели в виду создатели C. Второй цели идеально отвечает язык, который настолько «близок к поставленной задаче», что на нем непосредственно и точно выражаются понятия, используемые в решении задачи. Именно это имелось в виду, когда первоначально определялись средства, добавляемые к C. Связь между языком, на котором мы думаем и программируем, а также между задачами и их решениями, которые можно представить в своем воображении, довольно близка. По этой причине ограничивать возможности языка только поиском ошибок программиста - в лучшем случае опасно. Как и в случае естественных языков, очень полезно обладать, по крайней мере, двуязычием. Язык предоставляет программисту некоторые понятия в виде языковых инструментов; если они не подходят для задачи, их просто игнорируют. Например, если существенно ограничить понятие указателя, то программист будет вынужден для создания структур, указателей и т.п. использовать вектора и операции с целыми. Хороший проект программы и отсутствие в ней ошибок нельзя гарантировать только наличием или отсутствием определенных возможностей в языке. Типизация языка должна быть особенно полезна для нетривиальных задач. Действительно, понятие класса в C++ проявило

себя как мощное концептуальное средство.

Замечания о программировании на языке C++

Предполагается, что в идеальном случае разработка программы делится на три этапа: вначале необходимо добиться ясного понимания задачи, затем определить ключевые понятия, используемые для ее решения, и, наконец, полученное решение выразить в виде программы. Однако, детали решения и точные понятия, которые будут использоваться в нем, часто проясняются только после того, как их попытаются выразить в программе. Именно в этом случае большое значение приобретает выбор языка программирования.

Во многих задачах используются понятия, которые трудно представить в программе в виде одного из основных типов или в виде функции без связанных с ней статических данных. Такое понятие может представлять в программе класс. Класс - это тип; он определяет поведение связанных с ним объектов: их создание, обработку и уничтожение. Кроме этого, класс определяет реализацию объектов в языке, но на начальных стадиях разработки программы это не является и не должно являться главной заботой. Для написания хорошей программы надо составить такой набор классов, в котором каждый класс четко представляет одно понятие. Обычно это означает, что программист должен сосредоточиться на вопросах: Как создаются объекты данного класса? Могут ли они копироваться и (или) уничтожаться? Какие операции можно определить над этими объектами? Если на эти вопросы удовлетворительных ответов не находится, то, скорее всего, это означает, что понятие не было достаточно ясно сформулировано. Тогда, возможно, стоит еще поразмышлять над задачей и предлагаемым решением, а не немедленно приступать к программированию, надеясь в процессе него найти ответы.

Проще всего работать с понятиями, которые имеют традиционную математическую форму представления: всевозможные числа, множества, геометрические фигуры и т.д. Для таких понятий полезно было бы иметь стандартные библиотеки классов. Понятие не существует в вакууме, вокруг него всегда группируются связанные с ним понятия. Определить в программе взаимоотношения классов, иными словами, установить точные связи между используемыми в задаче понятиями, бывает труднее, чем определить каждый из классов сам по себе. В результате не должно получиться «каши» - когда каждый класс (понятие) зависит от всех остальных. Пусть есть два класса A и B. Тогда связи между ними типа «A вызывает функцию из B», «A создает объекты B», «A имеет член типа B№ обычно не вызывают каких-либо трудностей. Связи же типа «A использует данные из B», как правило, можно вообще исключить.

Одно из самых мощных интеллектуальных средств, позволяющих справиться со сложностью, - это иерархическое упорядочение, т.е. упорядочение связанных между собой понятий в древовидную структуру, в которой самое

общее понятие находится в корне дерева. Часто удается организовать классы программы как множество деревьев или как направленный ацикличный граф. Это означает, что программист определяет набор базовых классов, каждый из которых имеет свое множество производных классов. Набор операций самого общего вида для базовых классов (понятий) обычно определяется с помощью виртуальных функций. Интерпретация этих операций, по мере надобности, может уточняться для каждого конкретного случая, т.е. для каждого производного класса.

Естественно, есть ограничения и при такой организации программы. Иногда используемые в программе понятия не удается упорядочить даже с помощью направленного ацикличного графа. Некоторые понятия оказываются по своей природе взаимосвязанными. Циклические зависимости не вызовут проблем, если множество взаимосвязанных классов настолько мало, что в нем легко разобраться. Для представления на C++ множества взаимозависимых классов можно использовать дружественные классы.

Если понятия программы нельзя упорядочить в виде дерева или направленного ацикличного графа, а множество взаимозависимых понятий не поддается локализации, то, по всей видимости, выйти из этого положения не сможет помочь ни один из языков программирования. Если не удалось достаточно просто сформулировать связи между основными понятиями задачи, то, скорее всего, не удастся ее запрограммировать.

Еще один способ выражения общности понятий в языке предоставляют шаблоны типа. Шаблонный класс задает целое семейство классов. Например, шаблонный класс список задает классы вида «список объектов T», где T может быть произвольным типом. Таким образом, шаблонный тип указывает, как получается новый тип из заданного в качестве параметра. Самые типичные шаблонные классы – это контейнеры, в частности, списки, массивы и ассоциативные массивы. Однако весь аппарат построения новых типов следует привлекать только тогда, когда он действительно необходим.

Вопрос «Как написать хорошую программу на C++?» очень похож на вопрос «Как пишется хорошая английская проза?». На него есть два ответа: «Нужно знать, что вы, собственно, хотите написать» и «Практика и подражание хорошему стилю». Оба совета пригодны для C++ в той же мере, что и для английского языка, и обоим достаточно трудно следовать.

Несколько полезных советов

Ниже представлен «свод правил», который стоит учитывать при изучении C++. Сознательно выбраны очень простые правила, и в них опущены подробности. Не следует воспринимать их слишком буквально. Хорошая программа требует и ума, и вкуса, и терпения. С первого раза обычно она не получается, поэтому экспериментируйте! Итак, свод правил.

– Когда вы пишите программу, то создаете конкретные представления тех

понятий, которые использовались в решении поставленной задачи. Структура программы должна отражать эти понятия настолько явно, насколько это возможно.

- Если вы считаете «нечто» отдельным понятием, то сделайте его классом.
- Если вы считаете «нечто» существующим независимо, то сделайте его объектом некоторого класса.
- Если два класса имеют нечто существенное, и оно является для них общим, то выразите эту общность с помощью базового класса.
- Если класс является контейнером некоторых объектов, сделайте его шаблонным классом.
- Если определяется класс, который не реализует математических объектов вроде матриц или комплексных чисел и не является типом низкого уровня наподобие связанного списка, то: не используйте глобальных данных; не используйте глобальных функций (не членов); не используйте общих данных-членов; не используйте функции friend (но только для того, чтобы избежать вышеуказанное); не обращайтесь к данным-членам другого объекта непосредственно; не заводите в классе «поле типа»; используйте виртуальные функции; используйте функции-подстановки только как средство значительной оптимизации.

Замечание для программистов на C

Чем лучше программист знает C, тем труднее будет для него при программировании на C++ отойти от стиля программирования на C. Так он теряет потенциальные преимущества C++. Поэтому советуем просмотреть раздел «Отличия от C» в справочном руководстве. Здесь речь пойдет только о тех местах, в которых использование дополнительных возможностей C++ приводит к лучшему решению, чем программирование на чистом C. Макрокоманды практически не нужны в C++: используйте const или enum, чтобы определить поименованные константы; используйте inline, чтобы избежать расходов ресурсов, связанных с вызовом функций; используйте шаблоны типа, чтобы задать семейство функций и типов. Не описывайте переменную, пока она действительно вам не понадобится, а тогда ее можно сразу инициализировать, ведь в C++ описание может появляться в любом месте, где допустим оператор. Не используйте malloc(), эту операцию лучше реализует new. Объединения нужны не столь часто, как в C, поскольку альтернативность в структурах реализуется с помощью производных классов. Старайтесь обойтись без объединений, но если они все-таки нужны, не включайте их в основные интерфейсы; используйте безымянные объединения. Старайтесь не использовать указателей типа void*, арифметических операций с указателями, массивов в стиле C и операций приведения. Если все-таки вы используете эти конструкции, упрятывайте их достаточно надежно в какую-нибудь функцию или класс. Связывание в стиле C

возможно для функции на C++, если она описана со спецификацией extern "C".

Но гораздо важнее стараться думать о программе как о множестве взаимосвязанных понятий, представляемых классами и объектами, чем представлять ее как сумму структур данных и функций, что-то делающих с этими данными.