УЧЕБНО-НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС
«МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСТАНА»

«УТВЕРЖДЕНО»
Ректор НОУ УНПК «МУК»
к.т.н., доцент Савченков Е.Ю.
_____
«16» 10 2018 г.

БАКАЛАВРИАТ

Кафедра «Компьютерные информационные системы и управление»

Учебно-методический комплекс дисциплины

Технологии программирования

Направление: **710200 «Информационные системы и технологии»**

Профиль: **Информационные системы и технологии**

Академическая степень **- бакалавр**

Форма обучения **(очная)**

График проведения модулей 3-семестр

Нед.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Лек.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Лаб.	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

4-семестр

Нед.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Лек.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Лаб.	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

«РАССМОТРЕНО»
Протокол заседания кафедры
«КИСиУ»
№ 2 от 16.10.2018
Зав. кафедрой д.т.н., проф. Миркин Е.Л.
_____

«СОГЛАСОВАНО»
Проректор по академ. вопросам
проф. Мадалиев М.М.
_____

Составитель _____ к.т.н., и.о., доцента Нежинских С.С.

Директор Научной библиотеки _____ Асанова Ж.Ш.

БИШКЕК 2018

ОГЛАВЛЕНИЕ

АННОТАЦИЯ

Цель дисциплины (модулей) заключается в подготовке выпускника, имеющего специальные знания в области информационных технологий, для работы в области разработки программного обеспечения. Задачами дисциплины (модулей) являются: предоставление студентам навыков программирования на языке высокого уровня; ознакомление с основами методологии объектно-ориентированного программирования, основными алгоритмическими структурами, со структурным подходом к программированию; обучение основам программирования управления событиями, обработки исключительных событий, основам визуального программирования.

На изучение дисциплины отводится 180 часов в третьем семестре и 180 часов в четвёртом семестре. Рубежный контроль успеваемости проводится: в третьем семестре на 4, 8, 12, 16 неделях; в четвёртом семестре на 4, 8, 12, 15 неделях. Формы текущего контроля: опрос, проверка заданий, посещаемость. Форма рубежного контроля — модульная работа. Форма итогового контроля — экзамен.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ)

1. Пояснительная записка

1.1. Миссия и стратегия

Миссия НОУ УНПК "МУК" – подготовка международно - признанных, свободно мыслящих специалистов, открытых для перемен и способных трансформировать знания в ценности на благо развития общества.
Видение НОУ УНПК «МУК»- создание динамичного и креативного университета с инновационными научно-образовательными программами и с современной инфраструктурой, способствующие достижению академических и профессиональных целей.
Стратегии развития - модернизация образовательной деятельности университета – совершенствование образовательного процесса в соответствии с требованиями Болонского процесса.

1.2. Цель и задачи дисциплины (модулей)

Цель дисциплины: предоставление студентам навыков программирования на языке высокого уровня. Знакомство с современными технологиями разработки программного продукта в условиях многократного использования созданных программ и работы вычислительных систем в реальном масштабе времени.

Задачи дисциплины:

- получение знаний о современных методах в области программирования на языке C++;
- изучение современных конструкций языка программирования C++;
- получение навыков и приёмов визуального программирования;
- решение задач обработки данных;

1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине (модулю)

Дисциплина (модуль) направлена на формирование следующих компетенций:

- общенаучными (ОК):
  - способен использовать базовые положения математических / естественных / гуманитарных / экономических наук при решении профессиональных задач (ОК-2);
- профессиональными (ПК):
  - способен освоить методики использования программных средств для решения практических задач (ПК-2);
  - разрабатывать интерфейсы «человек - электронно-вычислительная машина» (ПК-3).

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

1. Знать:
   - концепции объектно-ориентированного программирования и механизмы их реализации в языке C++;
   - идеологию абстракции данных и абстрактных типов данных;
   - механизмы создания абстрактных типов данных;
   - принципы инкапсуляции данных в классах;
   - механизмы управления доступом к членам класса;
   - назначение дружественных функций и классов;
   - назначение статических членов класса;
   - механизмы создания и уничтожения объектов и манипулирования ими;
   - шаблоны классов;
   - идеологию наследования классов;
   - иерархию классов потоков ввода-вывода;
   - механизмы обработки исключительных ситуаций в языке C++.
2. Уметь:
   - переопределять операции для работы с новыми типами данных;
   - создавать объекты классов и работать с ними;
   - создавать и использовать шаблоны функций и классов;
   - создавать новые классы путем наследования существующих классов;
   - использовать виртуальные функции;
   - генерировать в программах исключительные ситуации и обрабатывать их.
3. Владеть:
   - навыками разработки алгоритмов, описания структур данных, описания основных базовых конструкций, программирования на языке высокого уровня.

1.4. Место дисциплины (модулей) в структуре ООП ВПО

Дисциплина (модуль) является частью общенаучного цикла (блока) дисциплин учебного плана по направлению подготовки 710200 «Информационные системы и технологии». Для освоения дисциплины (модулей) необходимы компетенции, сформированные в ходе изучения следующих дисциплин и прохождения практик: Математическая логика и теория алгоритмов, Основы программирования.

2. Структура и содержание дисциплины (модулей)

Структура дисциплины (модулей) для очной формы обучения
Общая трудоемкость дисциплины составляет: в третьем семестре 6 кредитов, 180 ч., в том числе аудиторная работа обучающихся с преподавателем 102 ч.,

самостоятельная работа обучающихся 78 ч; в четвёртом семестре 6 кредитов, 180 ч., в том числе аудиторная работа обучающихся с преподавателем 96 ч., самостоятельная работа обучающихся 84 ч.

№ п/п	*Раздел, Темы Дисциплины*	*Семестр*	*Неделя семестра*	*Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)*				*Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)*
				Лекции	*Сем /лаб*	*СРС*	*СРСП*	
Раздел 1.								
1	Функция. Определение собственных функций.	3	1	2	4	2	2	опрос, проверка задания, посещаемость
2	Собственные функции как отдельный файл.	3	2	2	4	2	2	опрос, проверка задания, посещаемость
3	Перегрузка функций. Функции с переменным числом параметров.	3	3	2	4	2	2	опрос, проверка задания, посещаемость
4	Указатели. Передача аргументов по ссылке. Массив в качестве аргументов функций.	3	4	2	4	2	2	Модуль 1
Раздел 2.								
5	Динамическое распределение памяти. Динамические массивы.	3	5	2	4	3	2	опрос, проверка задания, посещаемость

	Двумерные динамические массивы.							
6	Объектно-ориентированное программирование. Структуры данных. Объединения. Переменные структуры	3	6	2	4	3	2	опрос, проверка задания, посещаемость
7	Классы. Конструкторы и деструкторы.	3	7	2	4	3	2	опрос, проверка задания, посещаемость
8	Дружественные функции и их возможности.	3	8	2	4	3	2	Модуль 2
Раздел 3.								
9	Дружественные классы	3	9	2	4	3	2	опрос, проверка задания, посещаемость
10	Механизм наследования класса.	3	10	2	4	3	2	опрос, проверка задания, посещаемость
11	Определение производного класса.	3	11	2	4	3	2	опрос, проверка задания, посещаемость
12	Множественное наследование.	3	12	2	4	3	2	Модуль 3
Раздел 4.								
13	Использование указателей для доступа компонентам класса.	3	13	2	4	3	2	опрос, проверка задания, посещаемость
14	Виртуальные функции, раннее и позднее связывание.	3	14	2	4	3	2	опрос, проверка задания, посещаемость

15	Абстрактные классы.	3	15	2	4	3	2	опрос, проверка задания, посещаемость
16	Особенности объектно-ориентированного программирования.	3	16	2	4	3	2	Модуль 4
17	Консультация	3	17	2	4	0	2	посещаемость
Раздел 5.								
18	Приложение Windows Forms. Методы формы.	4	1	2	4	3	2	опрос, проверка задания, посещаемость
19	Компонент Button (кнопка). Виды кнопок (Button).	4	2	2	4	3	2	опрос, проверка задания, посещаемость
20	Компонент textBox (текстовое поле). Методы класса Convert (преобразование)	4	3	2	4	3	2	опрос, проверка задания, посещаемость
21	Объект String. Методы объекта String	4	4	2	4	3	2	Модуль 1
Раздел 6.								
22	Создание окон сообщений. Класс MessageBox.	4	5	2	4	3	2	опрос, проверка задания, посещаемость
23	Компоненты CheckBox (флажок) и RadioButton (переключатель)	4	6	2	4	3	2	опрос, проверка задания, посещаемость
24	Компонент ListBox. Компонент ComboBox (поле со списком).	4	7	2	4	3	2	опрос, проверка задания, посещаемость
25	Компонент PictureBox	4	8	2	4	3	2	Модуль 2

Раздел 7.								
26	Компонент Timer.	4	9	2	4	4	2	опрос, проверка задания, посещаемость
27	Передача данных между формами.	4	10	2	4	4	2	опрос, проверка задания, посещаемость
28	Мультимедиа возможности Windows Forms.	4	11	2	4	4	2	опрос, проверка задания, посещаемость
29	Хранение данных.	4	12	2	4	4	2	Модуль 3
Раздел 8.								
30	Параллельные вычисления.	4	13	2	4	4	2	опрос, проверка задания, посещаемость
31	Утилизация неиспользуемых объектов.	4	14	2	4	4	2	опрос, проверка задания, посещаемость
32	Эффективное использование памяти приложением.	4	15	2	4	4	2	Модуль 4
33	Консультация	4	16	2	4	0	2	посещаемость

3. Конспект лекций

Конспект лекций можно просмотреть в приложении 1.

4. Информационные и образовательные технологии

№ п/п	Наименование раздела	Виды учебной работы	Формируемые компетенции (указывается код компетенции)	Информационные и образовательные технологии
1	Разделы: 1, 2, 3, 4, 5, 6, 7, 8	Лекция	ОК-2, ПК-2, ПК-3	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ОК-2, ПК-2, ПК-3	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ОК-2, ПК-2, ПК-3	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты

5. Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоению дисциплины (модулей)

5.1. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины

№ п/п	Контролируемые разделы дисциплины (модулей)	Код контролируемой компетенции (компетенций)	Наименование оценочного средства
1	Разделы №1, №2, №3, №4, №5, №6, №7, №8	ОК-2	опрос, выполнение лабораторных работ
2	Разделы №1, №2, №3, №4, №5, №6, №7, №8	ПК-2	опрос, выполнение лабораторных работ
3	Разделы №1, №2, №3, №4, №5, №6, №7, №8	ПК-3	опрос, выполнение лабораторных работ

5.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

5.2.1. Третий семестр

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 1, 2, 3, 4 недели 1, 2, 3, 4 недели 1, 2, 3, 4 недели	 8 баллов 10 баллов 2 балла	 До 32 баллов До 40 баллов 8 баллов
Рубежный контроль: (сдача модуля)	4 неделя	100%×0,2=20 баллов	
Итого за I модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего

Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 5, 6, 7, 8 недели 5, 6, 7, 8 недели 5, 6, 7, 8 недели	 8 баллов 10 баллов 2 балла	 До 32 баллов До 40 баллов 8 баллов
Рубежный контроль: (сдача модуля)	8 неделя	100%×0,2=20 баллов	
Итого за II модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 9, 10, 11, 12 недели 9, 10, 11, 12 недели 9, 10, 11, 12 недели	 8 баллов 10 баллов 2 балла	 До 32 баллов До 40 баллов 8 баллов
Рубежный контроль: (сдача модуля)	12 неделя	100%×0,2=20 баллов	
Итого за III модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ	 13, 14, 15, 16 недели 13, 14, 15, 16 недели	 8 баллов 10 баллов	 До 32 баллов До 40 баллов

- посещаемость	13, 14, 15, 16 недели	2 балла	8 баллов
Рубежный контроль: (сдача модуля)	16 неделя	100%×0,2=20 баллов	
Итого за IV модуль			До 100 баллов
Итоговый контроль (экзамен)	Сессия	ИК = Бср × 0,8 + Бэкз × 0,2	

Полученный совокупный результат (максимум 100 баллов) конвертируется в традиционную шкалу:

Рейтинговая оценка (баллов)	**Оценка экзамена**
от 0 до 54	неудовлетворительно
от 55 до 69	удовлетворительно
от 70 до 84	хорошо
от 85 до 100	отлично

5.2.2. Четвёртый семестр

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	1, 2, 3, 4 недели	8 баллов	До 32 баллов
- выполнение лабораторных работ	1, 2, 3, 4 недели	10 баллов	До 40 баллов
- посещаемость	1, 2, 3, 4 недели	2 балла	8 баллов
Рубежный контроль: (сдача модуля)	4 неделя	100%×0,2=20 баллов	
Итого за I модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 5, 6, 7, 8 недели 5, 6, 7, 8 недели 5, 6, 7, 8 недели	 8 баллов 10 баллов 2 балла	 До 32 баллов До 40 баллов 8 баллов
Рубежный контроль: (сдача модуля)	8 неделя	100%×0,2=20 баллов	
Итого за II модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 9, 10, 11, 12 недели 9, 10, 11, 12 недели 9, 10, 11, 12 недели	 8 баллов 10 баллов 2 балла	 До 32 баллов До 40 баллов 8 баллов
Рубежный контроль: (сдача модуля)	12 неделя	100%×0,2=20 баллов	
Итого за III модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль:			

- опрос - выполнение лабораторных работ - посещаемость	13, 14, 15 недели 13, 14, 15 недели 13, 14, 15 недели	7 баллов 15 баллов 4 балла	До 21 баллов До 45 баллов 12 баллов
Рубежный контроль: (сдача модуля)	16 неделя	100%×0,2=20 баллов	
Итого за IV модуль			До 100 баллов
Итоговый контроль (экзамен)	Сессия	ИК = Бср × 0,8 + Бэкз × 0,2	

Полученный совокупный результат (максимум 100 баллов) конвертируется в традиционную шкалу:

Рейтинговая оценка (баллов)	**Оценка экзамена**
от 0 до 54	неудовлетворительно
от 55 до 69	удовлетворительно
от 70 до 84	хорошо
от 85 до 100	отлично

5.3. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Текущий контроль (0-80 баллов)

При оценивании посещаемости, опроса и выполнении лабораторных работ учитываются:

- посещаемость (10 баллов)
- степень раскрытия содержания материала (25 баллов);
- изложение материала (грамотность речи, точность использования терминологии и символики, логическая последовательность изложения материала (20 баллов);
- знание теории изученных вопросов, сформированность и устойчивость используемых при ответе умений и навыков (25 баллов).

Рубежный контроль (0 – 20 баллов)

При оценивании контрольной работы учитывается:

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – 10 баллов;
- обоснованность содержания и выводов работы (задание выполнено полностью, но обоснование содержания и выводов недостаточны, но рассуждения верны) – 5 баллов;
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок, возможна одна неточность – 5 баллов.

Итоговый контроль (экзаменационная сессия) – ИК = Бср × 0,8 + Бэкз × 0,2

При проведении итогового контроля обучающийся должен ответить на 3 вопроса (два вопроса теоретического характера и один вопрос практического характера).

При оценивании ответа на вопрос теоретического характера учитывается:

- теоретическое содержание не освоено, знание материала носит фрагментарный характер, наличие грубых ошибок в ответе (0 баллов);
- теоретическое содержание освоено частично, допущено не более двух-трех недочетов (10 баллов);
- теоретическое содержание освоено почти полностью, допущено не более одного-двух недочетов, но обучающийся смог бы их исправить самостоятельно (20 баллов);
- теоретическое содержание освоено полностью, ответ построен по собственному плану (30 баллов).

При оценивании ответа на вопрос практического характера учитывается:

- ответ содержит менее 20% правильного решения (0-9 баллов);
- ответ содержит 21-89 % правильного решения (10-39 баллов);
- ответ содержит 90% и более правильного решения (40 баллов).

6. Учебно-методическое и информационное обеспечение дисциплины

6.1. Список источников и литературы

Литература:

- Основная:

  1. Холзнер С. Visual C++ 6: учебный курс - СПб: Питер,2001. - 576 с.
  2. Тихомиров Ю. Visual C++ 6 - Спб.:БХВ - Санкт-Петербург, 1998. - 496 с.
  3. Дейтел Х., Дейтел П. Как программировать на C++: Пер. с англ. - М,: Издательство БИНОМ, 1998 - 1024 с.
  4. Шилдт, Герберт. Полный справочник по C, 4-е издание. : Пер. с англ. -

М,: Издательский дом "Вильямс", 2002. - 704 с.
5. Шилдт, Герберт. Самоучитель C++, 3-е издание: пер. с англ. - СПб.: BVH - Санкт-Петербург, 1998.-688 с.
6. Павловская Т.А. С/С++. Программирование на языке высокого уровня. / Т.А. Павловская. - СПб.: Питер, 2002. - 464 с.
7. Культин Н.Б. С/С++ в задачах и примерах. - СПб.:БХВ-Петербург, 2001. - 288 с.

- Дополнительная:
1. Березин Б.И., Березин С.Б. Начальный курс С и С++. - М,: ДИАЛОГ_МИФИ, 1996. - 288 с.
2. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учеб. пособие. - М.: Финансы и статистика, 1998. - 600 с.
3. Франка П. С++: учебный курс. - СПб.: Питер, 2001. - 528 с.
4. Дэвис Стефан Р. С++ для "чайников", 4-е издание.: Перев. с англ.: Уч. пос. - М.: Издательский дом "Вильямс", 2001. - 336 с.
5. Джонс Р., Стюарт Я. Программируем на Си/Пер. с англ. и предисл. М.Л. Сальникова, Ю.В. Сальниковой. - М.: Компьютер, ЮНИТИ, 1994. - 236 с.
6. Скляров В.А. Программирование на языках Си и Си++: Практ. пособие. - М.: Высш. шк., 1996. -240 с.
7. Пашенков В.В. Язык программирования Си. - М.: Центр НТТМ "Алгоритм", 1990. - 76 с.
8. УинерР Р. Язык Турбо Си: Пер. с англ. -М.: Мир, 1991. - 384 с.: ил.
9. Першиков В.И., Савинков В.М. Толковый словарь по информатике.-М.: Финансы и статистика, 1991.-543 с.
10. Киммел П. и др. Borland C++ 5: пер. с англ. - Санкт-Петербург, 1997.- 976 с.

6.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей)

- Visual C++ Documentation: https://docs.microsoft.com/en-us/cpp/
- http://www.iprbookshop.ru/
- http://kyrlibnet.kg/ru/
- http://biblioteka.kg/

7. Материально-техническое обеспечение дисциплины

Для изучения дисциплины, необходимо следующее оборудование: ЭВМ, проектор.

Требования к аудитории: компьютерный класс, имеющий ЭВМ в количестве

идентичном количеству обучающихся, ЭВМ для преподавателя с подключенным проектором, наличие доски и средств для отображения/удаления информации на доске (мел/ветошь, маркер/губка).

8. Приложения

Приложение 1

Лекция 1. Приложение Windows Forms
Форма – это главный контейнер, в котором размещаются компоненты самой среды. С помощью этих компонентов реализуется конкретный алгоритм, определенной задачи. При написании программного кода, все обработчики событий формируется в рамках одного класса формы.
Поэтому обращение к составляющим класса осуществляются, через указатель на экземпляр этого класса, то есть указатель this.
Указатель this всегда содержит ссылку на текущий объект.
Рассмотрим некотырые методы формы.

1. Close() – метод закрывающий форму.
2. Hide() – метод скрывающий форму, то есть форма становится невидимой.
3. Show() – метод выводит форму на экран.
4. ShowDialog() – показывает форму в модальном режиме.

Если в программе необходимо использовать несколько форм, то для каждой новой формы необходимо добавить свой класс формы и в главной форме подключить заголовочные файлы.

Код:

```
t1^newDlg1=gcnew t1();
newDlg1->Show();

t2^newDlg2=gcnew t2();
newDlg2->Show();

this->Close(); для закрытия формы
#include "t1.h"
#include "t2.h"
```

^ – указатель
gcnew – new создание нового объекта
§ 1.1 Компонент Button (кнопка).
Создает в форме элемент кнопка.
Рассмотрим некоторые методы компонента Button:

1. Hide() – скрывает кнопку.
2. Focus() – делает кнопку активной.
3. Show() – показывает кнопку.

§ 1.2 Компонент Label (метка, надпись).
Этот компонент выводит в свое поле текст, который пользователь в режиме исполнения приложения не может редактировать.
Компонент используется, чтобы идентифицировать некоторый объект в форме. Данный компонент не может получать фокус ввода.

Лекция 2. Компонент textBox (текстовое поле)
Этот компонент задает в форме однострочное или многострочное редактируемое поле. Через которое вводят или выводят строчные данные.
Рассмотрим некоторые методы компонента textbox:

1. AppendText() – добавляет текст к текущему тексту в окне компонента.
2. Clear() – удаляет весь текст из поля.
3. Focus() – устанавливает фокус ввода.
4. Hide() – скрывает компонент.
5. Show() – показывает компонент.

По умолчанию все данные в текстовом поле имеют тип String. Для преобразование в другие типы данных, используются методы класса Convert (преобразование):

1. Convert :: ToSingle() – число с плавающей точкой.
2. Convert :: ToDouble() – число с плавающей точкой.
3. Convert :: ToInt32() – целое число
4. Convert :: ToString() – в строку

Рассмотрим пример:
Пример: Найти сумму чисел введенных в первое и второе текстовое поле. Результат вывести в третье текстовое поле, после нажатия кнопки.

Код для нахождения сложения:

```
                    float x,y,z;
 x=Convert::ToSingle(textBox1->Text);
y=Convert::ToSingle(textBox2->Text);
z=x+y;
textBox3->Text=Convert::ToString(z);
```

Лекция 3. Объект String
Объект String предназначен для работы со строками. Для объявления строковой переменной необходимо использовать следующую запись:

```
String ^x;
```

Кроме того строковую переменную можно сразу инициализировать. Пример:

```
String ^y="Привет";
```

Для определения длины строки используется свойство length.

```
int i;
i=y->length;
```

Рассмотрим некоторые методы класса String:

1. Insert() – добавляет подстроку начиная с указанного индекса. Пример:
x=y->Insert(3,"А"); // x="ПриАвет";

1. IndexOf() – находит подстроку и возвращает индекс первого вхождения подстроки. Пример:
i=y->IndexOf("вет");// i=3;
i=y->IndexOf("Пр",3); начинать с заданной позиции
Если метод ничего не находит возвращает -1.

1. LastIndexOf() – находит под строку и возвращает индекс последнего вхождения в текст. Пример:
y="Колобок";
i=y->LastIndexOf("о"); // i=5;

1. Remove() – удаляет заданное количество символов из строки. Пример:
x=y->Remove(3,2);

1. Substring() – выделяет подстроку в строке.
y="Колобок";
x=y->Substing(4);// x="бок"
x=y->Substing(4,1);

```
int s=0;
String ^x;
x=textBox1->Text;
for(int i=0; i<x->Length; i++)
{
if(x->Substring(i,1)==" "||x->Substring(i,1)=="\n")
s++;
}
s=s+1;
textBox2->Text=Convert::ToString(s);
```

Лекция 4. Создание окон сообщений
Для создания окон сообщений используется метод Show класса MessageBox.
Синтаксис метода:
MessageBox:: Show (text, caption, buttons, icon);
text - это текст отображаемый в окне сообщений.
caption - это текст отображаемый в строке заголовка сообщения.
buttons – это одно из значений «MessageBoxbuttons» указывающие, какие кнопки отображаются в окне сообщения.
icon– это одно из значений «MessageBoxicon» указывающий, какой значок отображается в окне сообщения.
Английский
Виды кнопок
OK

OKCancel
YesNo
Yes No Cancel
Retry Cancel
AbortRetryIgnoreпрервать, повторить, игнорировать. При нажатии на одну из этих кнопок возвращается значение.
Dialog Result:: Yes
Типы icon

1. Отсутствие icon –None.
2. Вопросительный тип Question
3. Критический тип Hand, Stop, Error.
4. Предупредительный тип Exclamation, Warning.
5. Информационный тип Asterisk, Information.

Код:

```
MessageBox::Show("Привет","Нит", MessageBoxButtons::OK, MessageBoxIcon::Information);

MessageBox::Show("Привет \n всему \n НИТу" );
```

Рассмотрим пример: Найти корни квадратного уровнения.

```
double a,b,c,x1,x2,d;
a=b=c=x1=x2=d=0;
a=Convert::ToDouble(textBox1->Text);
b=Convert::ToDouble(textBox2->Text);
c=Convert::ToDouble(textBox3->Text);
d=b*b-4*a*c;
if(d<0)
{
MessageBox::Show("Корней нет");
}
else if(d==0)
{
x1=(-b)/(2*a);
MessageBox::Show("x="+Convert::ToString(x1));
}
else
{
x1=0.5*(-b+sqrt(d))/a;
x2=0.5*(-b-sqrt(d))/a;
MessageBox::Show("x1="+Convert::ToString(x1)+"\r\n"+
"x2="+Convert::ToString(x2));
}
```

Лекция 5. Компоненты CheckBox (флажок) и RadioButton (переключатель)
Оба компонента используются для предоставления выбора и выдают результат «включен или выключен».
Если компонент CheckBox (флажок) находится в группе себе подобных, то включив один флажок можно включить и остальные, при этом не один из них не выключится.
Если компонент RadioButton (переключатель) находится совместно с такими же компонентами в одной группе, то не допускает, чтобы одновременно был включен другой переключатель.
Для создания группы используется компонент groupBox.
Пример: Разработать форму, в которой в зависимости от выбранного переключателя будут появляться соответствующее изображение.

Рис 1.

Код:

```
if(radioButton1->Checked)
pictureBox1->ImageLocation="red.jpg";
if(radioButton2->Checked)
pictureBox1->ImageLocation="green.jpg";
if(radioButton3->Checked)
pictureBox1->ImageLocation="blue.jpg";
```

Пример№2:

Рис 2.
Код:

```
String ^f, ^i, ^v, ^l;
f=textBox1->Text;
i=textBox2->Text;
v=l=" ";
if(radioButton1->Checked)
v="МУК";
if(radioButton2->Checked)
v="КНУ";
if(radioButton3->Checked)
    v="КРСУ";
if(checkBox1->Checked)
l+="русский,";
if(checkBox2->Checked)
l+="кыргызский,";
if(checkBox3->Checked)
```

```
l+="английский,";
if(checkBox4->Checked)
l+="немецкий,";
textBox3->Text=f+" "+i+"\n"+v+"\n"+l;
```

Лекция 6. Компонент ListBox

listBox – предназначен для создания списков. Если элементы списка превосходят размеры окна списка, автоматически появляется полоса прокрутки.

Рассмотрим некоторые методы компонента.

1. Add ( ) добавляет элементы в конец списка.
2. Insert ( ) добавляет элемент внутрь списка.
3. Clear ( ) очищает список, то есть удаляет все элементы списка.
4. Remove ( ) удаляет заданный элемент из списка.

Например:

```
listBox1->Items->Add("строка1");
listBox1->Items->Add("строка2");

listBox1->Items->Insert(0,"строка3");

listBox1->Items->Remove("строка1");

listBox1->Items->Clear();
```

Items – это элемент списка их можно просматривать или изменять.

1. Извлечение строки из списка.

```
String ^x;
        x=listBox1->Items[1]->To string();
textBox1->Text=x;
```

1. Извлечение строки из списка, на которой был щелчок мыши.

```
String ^x;
x=listBox1->Items[listBox1->SelectedIndex]->To string();
textBox1->Text=x;
```

Пример:

Код:

```
listBox1->Items->Add("программист");
listBox1->Items->Add("сисадмин");
listBox1->Items->Add("дворник");
listBox1->Items->Add("повар");
String ^f, ^i, ^l, ^p;
```

```
f=textBox1->Text;
i=textBox2->Text;
l=" ";
if(checkBox1->Checked)
l+="русский,";
if(checkBox2->Checked)
l+="кыргызский,";
if(checkBox3->Checked)
l+="английский,";
if(checkBox4->Checked)
l+="немецкий,";
p=listBox1->Items[listBox1->SelectedIndex]->ToString();
textBox3->Text=f+" "+i+"\n"+l+"\n"+p;
}
```

Лекция 7. Компонент ComboBox (поле со списком)

Компонент является комбинацией редактируемого поля и списка, используется для вывода данных в виде выпадающего списка и последующей выборки их из этого списка.

Свойство:

1. Items – содержит набор строк.
2. SelectedIndex – целочисленная переменная индекс выбранной строки, если элементы не выбраны возвращается значение -1.
3. SelectedItem – возвращает выбранный элемент.
4. Count – возвращает количество элементов в списке.

```
int x=comboBox1->Items->Count
```

1. Text – содержит значение поля для редактирования компонента, тоесть извлекает строку.

Методы:

1. Clear – очищает компонент.
2. Add – добавляет элемент в конец списка.
3. Indexof() –возвращает индекс строки, строка задается в аргументе метода, если строка не найдена возвращается -1.
4. Insert () – вставляет строку внутри списка.
5. Remove () – удаляет заданную строку.
6. RemoveAdd () – удаляет требуемую строку в аргументе,указывается индекс строки.

Лекция 8. Компонент pictureBox

Через данный компонент в форму выводится графическое изображение.

Свойство:

1. Image – задает изображение загружаемое в компонент.
2. ImageLocation – указывает путь загружаемого изображения.

```
pictureBox1->ImageLocation="1.jpg";
```

Метод:
Load() – позволяет загрузить требуемый рисунок.
Пример:

Код:

```
String ^x;
x=textBox1->Text;
pictureBox1->Load(x);

int x,y;
x=Convert::ToInt32(textBox1->Text);
y=Convert::ToInt32(textBox2->Text);
array<String^>^t=gcnew array<String^>(4);// создание массива строк из 4 элементов
String ^a;
t[0]=pictureBox1->ImageLocation;
t[1]=pictureBox2->ImageLocation;
t[2]=pictureBox3->ImageLocation;
t[3]=pictureBox4->ImageLocation;
a=t[x-1];
t[x-1]=t[y-1];
t[y-1]=a;
pictureBox1->ImageLocation=t[0];
pictureBox2->ImageLocation=t[1];
pictureBox3->ImageLocation=t[2];
pictureBox4->ImageLocation=t[3];
```

Лекция 9. Компонент Таймер
Компонент таймер создает счетчик времени. Является удобным средством для организации процессов, автоматически повторяющихся через равные интервалы времени.
Свойство:

1. Enabled – управляет запуском и остановкой таймера.
2. Interval – задает промежуток времени через который возникает единственное его событие Tick.

Методы компонента:

1. Start() – запускает таймер
2. Stop() – останавливает таймер

Рассмотрим пример: Создать программу, которая случайным образом генерирует массив из пяти однозначных чисел. Необходимо запустить таймер

для чередования графических изображений этих чисел с интервалом 10 сек.

Код:

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^ e) {
timer1->Start();
timer1->Interval=10000;
}
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^ e) {
timer1->Stop();
}
private: System::Void timer1_Tick(System::Object^  sender, System::EventArgs^  e) {
int x=0;
Random^y=gcnew Random;
x=y->Next(1,6);
label1->Text=" "+x;
if(x==1)
pictureBox1->ImageLocation="1.jpg";
if(x==2)
pictureBox1->ImageLocation="2.jpg";
if(x==3)
pictureBox1->ImageLocation="3.jpg";
if(x==4)
pictureBox1->ImageLocation="4.jpg";
if(x==5)
pictureBox1->ImageLocation="5.jpg";
}
```

Лекция 10. Компонент DataTimePicker

Создает календарь. При выборе даты или времени компонент может представляться в двух форматах: в виде прямоугольного поля, в котором высвечивается дата или время и в виде выпадающего списка с датами.

Дата и время относятся к типу DataTime, котрый имеет специальный механизм позволяющий предоставлять эти две величины в виде строки.

Рассмотрим пример:

```
DataTime^t=DataTime();
String^s;
t=DataTimePicker1->Value;
s=t->ToString("dd/mm/yyyy");
textBox1->Text=s;
s=t->ToString("d mmm yyyy\r\,ddd");//31 март 2014г понедельник
```

Свойство компонента:

1. Day – день
2. Month – месяц
3. Year – год
4. Hour – час
5. Minute – минута
6. Second – секунда

int x=t->Day;// Month

Лекция 11. Компонент ProgressBar

Создает индикатор процесса, благодаря чему можно наблюдать ход процесса во времени. Прямоугольный индикатор при достаточно длинном процессе постоянно заполняется символом заполнения слева на право. Причем заполнение завершается с окончанием самого процесса. Это заполнение организовано с помощью свойств и методов компонета ProgressBar.

Свойство:

1. Min Max – свойства задают интервал значений индикатора.
2. Value – определяет текущую позицию индикатора, внутри интервала.

Методы:

1. Perform(n - целое число) – задает шаг при решение равное n.
2. PerformStep() – вызывает изменение свойства Value на величину приращения.

Рассмотрим пример:

Код:

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {
timer1->Start();
timer1->Interval=1000;
progressBar1->Minimum=0;
progressBar1->Maximum=10;
progressBar1->Value=0;
}
private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
timer1->Stop();
MessageBox::Show("Процесс завершен");
progressBar1->Visible=false;
}
private: System::Void timer1_Tick(System::Object^ sender, System::EventArgs^ e)
{
progressBar1->Value++;
if(progressBar1->Value==progressBar1->Maximum)
button2_Click(sender,e);
}
```

Лекция 12. Компонент MenuStrip и TabControl

Компонент MenuStrip создает главное меню приложения, с помощью которого управляют всей работой приложения и его частей.

Компонент TabControl позволяет создавать вкладки. Каждая вкладка это отдельный объект TabPage со своими свойствами, методами и событиями. При перемещении с одной страницы на другую при щелчке мыши возникает событие Click для объекта TabPage.

Некоторые свойства:

1. SelectedIndex – содержит номер активной страницы, если не одна страница не выбрана возвращает.
2. SelectedTab – активирует нужную вкладку.

Пример: tabControl1->SelectedTab(2);

Лекция 13. Управление исключительными ситуациями

В процессе исполнения приложений могут возникнуть так называемые исключения (исключительные ситуации).

Исключительные ситуации это аномальное явление, которые требуют от программы определенной реакции.

Например: 1) деление на 0 превышение размерности массива.

В этом случае программист как то реагирует, но отклонение от нормального режима, либо система отреагирует сама, то есть обработает по своему такую ситуацию, выдаст сообщение и завершит работу программы.

В таких случаях желательно не дожидаться реакции системы и самому предпринимать меры, то есть управлять проблемами возникающими в процессе выполнения программы.

Язык C++ обеспечивает встроенную поддержку для обработки возникающих исключений. Управление обработкой исключений состоит в реакции программы на возникновение, неожидаемых событий существует особый класс.

System::Exceptions - содержащий необходимые свойство и методы помогающие обрабатывать исключительные ситуации.

Свойство класса:

1. Message – содержит описание возникшего исключения.
2. Source – содержит имя источника, где возникла ошибка
3. Helplink – содержит ссылку на соответствующий файл описывающий возникающую ситуацию.
4. HResult – содержит числовой код возникающей ошибки.

Рассмотрим операторы:

try, catch, throw – С помощью этих операторов обрабатываются исключительные ситуации.

Рассмотрим синтаксис:

```
try
{
участок программы, в котором могут возникнуть исключительные ситуации.
```

}
catch (объявление исключения)
{
опреаторы для обработки возникшего в try - блоке исключения.
}
throw выражение;
Оператор throw выбрасывает, для последующей обработки возникающего исключения.
Тело оператора catch является обработчиком исключения, то есть исключение выбрасывается оператором throw, захватывается оператором catch и в его теле обрабатывается.
Объявления исключения в операторе catch определяет тип исключения, которое станет обрабатываться в catch теле. Если в объявлении исключения задано многоточие, то это означает, что оператор catch станет обрабатывать любой тип возникающего исключения. Такой catch всегда записывается последним.
При создании обработки исключений блоков catch может быть более одного. Если необходимо отлавливать конкретные исключения нужно сформулировать для каждого из них свой catch с задание типа обрабатываемого исключения.
Рассмотрим алгоритм обработки исключительной ситуации:

1. Система исполняет операторы в обычном режиме. Не обращая внимания на наличие операторов обработки исключительных ситуаций.
2. Как только в момент исполнения участка программы, в теле try блока возникает исключительная ситуация, то создается объект класса exceptions и отыскивается наивысший по иерархии оператор catch, который может обработать исключение возникшего типа. Если подходящего по возникшему типу исключения оператора catch не находится, то отыскивается ближайший, следующий try-блок. Этот процесс продолжается пока самый краткий блок не повторится.
3. Если подходящий try-блок не будет обнаружен вызывается специальная функция среды, которая завершает работу приложения. Если же подходящий try-блок найден, то начинается обработки исключения соответствующим оператором catch.

Пример:

```
try
{
x=Convert::ToDouble(textBox1->Text);
y=Convert::ToDouble(textBox2->Text);
z=x/y;
textBox3->Text=Convert::ToString(z);
}
catch
{
MessageBox::Show("Деление на ноль");
```

}

Лекция 14. Фундаментальные характеристики Объектно-ориентированного программирования (ООП)

1. Все является объектом.

2. Вычисления осуществляются путем взаимодействия между объектами. При котором один объект требует, чтобы другой объект выполнил некие действия. Объекты взаимодействуют, посылая и получая сообщения. Сообщение это запрос на выполнения действия, дополненный набором аргументов, которые могут понадобится при выполнении действия.

3. Каждый объект является представителем класса, который выражает общие свойства объектов.

4. В классе задается поведение объекта, тем самым все объекты, которые являются экземплярами одного класса, могут выполнять одни и те же действия.

5. Классы организованны в единую древовидную структуру с общим корнем называемую иерархией наследования. Память и поведение связанное с экземплярами единого класса автоматически доступны любому классу расположенному ниже в иерархическом дереве.

Таким образом ООП рассматривает программы как совокупность свободно связанных между собой объектов. Каждый из них отвечает за конкретные задачи. Вычисление осуществляется взаимодействия объектов. Поведение объекта диктуется его классом.

Каждый объект является экземпляром некоторого класса. Объект проявляет свое поведение путем вызова метода в ответ на сообщение.

Разновидность классов:

Классы в ООП имеют несколько различных форм и используются для различных целей.

Следующие категории охватывают большую часть классов:

- Управление данными
- Источники данных
- Классы для просмотра данных
- Вспомогательные классы

1. Классы администраторы данных (управление) – это классы, основной особенностью которых является поддержка данных или предоставлении информации о состоянии чего либо. Классы администрирования данных обычно являются фундаментальными, строительными блоками проекта.

2. Источники данных – это классы, которые генерируют данные. Например: класс Random, также существуют посредники при передаче данных, которые служат для приема и дальнейшей передачи данных. В отличие от администраторов данных источники и посредники не хранят внутри себя данные в течении неопределенного времени, но генерирует их по запросу ил обрабатывают их при вызове.

3. Классы для просмотра данных – осуществляют вывод информации.

4. Вспомогательные классы – к вспомогательным классам относятся те

классы, которые не содержат полезной информации, но облегчает выполнение сложных заданий.

Наследование: преимущества и недостатки.

Преимущества:

1. Повторное использование программ
2. Использование общего кода
3. Согласование интерфейса
4. Программные компоненты
5. Быстрое макетирование
6. Полиморфизм
7. Маскировка информации (инкопсуляция)

Недостатки:

1. Скорость выполнения
2. Размер программ
3. Сложность программ

Лекция 15. Формы наследования

1. Порождение подкласса для специализации – при порождение подкласса для специализации, новый класс является специализированной формой родительского класса, но удовлетворяет спецификациям родителя во всех существенных моментах. Таким образом, дочерний класс является более конкретным, частным или специализированным случаем родительского класса. Другими словами дочерний класс является подтипом родительского класса.

2. Порождения подкласса для спецификации – данная форма наследования состоит в том, чтобы гарантировать поддержку классами определенного, общего интерфейса то есть, реализацию или одних и тех же методов. Родительский класс может быть комбинацией реализованных операций и операций осуществление, которых доверено дочерним классам.

Часто нет никаких отличий в интерфейсе между родительским и дочерними классами. Последний просто обеспечивает выполнения описанного, но не реализованного в родительском классе поведения. Таким образом, родительский класс описывает поведение которое реализуется в дочернем классе, но остановлено не реализованным в родительском.

1. Порождение подкласса с целью конструирования – в этом случае класс наследует почти все функциональное поведение родительского класса, изменяя только имена методов или определенным образом модифицируя аргументы.

Таким образом, дочерний класс использует методы, предоставляемые родительским классом. Но не является подтипом родительского класса.

1. Порождение подкласса для обобщения – использование наследования при порождении подклассов для обобщения в определенном смысле является противоположностью порождению для специализации. В этом случае подкласс расширяет родительский класс для создание объекта более общего типа. Порождение подкласса для обобщения часто применяется когда построение происходит на основе существующих классов, которые мы не хотим или не

можем изменять. Таким образом, дочерний класс модифицирует или переопределяет некоторые методы родительского класса с целью получения объекта более общей категории.

2. Порождение подкласса для расширения – в то время как порождения подкласса для обобщения модифицирует или расширяет существующие функциональные возможности объекта, порождение для расширения добавляет совершенно новые свойства. Расширение добавляет новые методы к родительским и в функциональные возможности подкласса менее крепко привязаны к существующим методам родителя.

Таким образом, дочерний класс добавляет новые функциональные возможности, к родительскому классу, но не меняет наследуемое поведение.

1. Порождение подкласса для ограничения – порождение для ограничения происходит в случае, когда возможности подкласса более ограничены, чем в родительском классе. Порождение для ограничения чаще всего возникает, когда программист, строит класс на основе существующей иерархии, которая не должна или не может быть изменена. Таким образом, дочерний класс ограничивает использование некоторых методов родительского класса.

2. Порождение подкласса для варьирования – порождение подкласса для варьирования, применяется, когда два класса имеют сходную реализацию, но не имеют никакой видимой иерархической связи. Таким образом, дочерний и родительские классы являются вариациями на одну тему и связь, класс подкласс произвольна.

3. Порождение подкласса для комбинирования – порождение подкласса для комбинирования является множественным наследованием. Таким образом, дочерний класс наследует черты более чем одного родительского класса.

Лекция 16. Программирование в малом и в большом

О разработке индивидуального проекта часто говорят, как о программировании «в малом», а о реализации «в большом».

Для программирование в малом характерны следующие признаки:

1. Программный код разрабатывается единственным программистом, если возможно небольшой группой программистов.

Отдельно взятый индивидуум может понять все аспекты проекта от и до.

1. Основная проблема при разработке состоит в проектировании программы и написании алгоритмов для решения поставленной задачи.

Программирование «в большом» наделяет продукт следующими свойствами:

1. Программная система разрабатывается большой командой программистов. При этом одна группа может заниматься проектированием системы, другая осуществлять написание кода отдельных компонентов, а третья объединять фрагмент конечный продукт. В этом случае нет единственного человека, который знал бы все о проекте.

2. Основная проблема в процессе разработки программного обеспечения – это управление проектом и обмен информацией между группами и внутри групп.

Лекция 17. Основные этапы проектирования приложения

1. Идентификация компонент

Разработка программного обеспечения облегчается после выделения отдельных компонент программы. Компонента – это абстрактная единица, которая может выполнять определенную работу. На этом этапе нет необходимости знать в точности как задается компонента или как она будет выполнять свою работу.

Компонента может в конечном итоге быть преобразована в отдельную функцию структуру или класс, или совокупность других компонент. На этом уровне в разработки имеются две важные особенности:

1. Компонента должна иметь небольшой набор четко определенных обязанностей;

2. Компонента должна взаимодействовать с другими компонентами настолько слабо, насколько это возможно. Что бы выявить отдельные компоненты команда программистов прорабатывает сценарий системы, то есть воспроизводиться запуск приложения, как если бы оно было уже готово. Любое действие, которое может произойти, приписывается некоторой компоненте в качестве ее обязанности, то есть выделение компонент производится во времени процесса мысленного представления работы системы.

Часто это происходит как цикл вопросов что? или кто? То есть команда программистов определяет что требуется делать.

1. Документирование

На этом этапе следует начать разработку документации, два документа должны являться существенными составными частями любой программной системы. Это руководство пользователя и проектная документация системы. Работа над каждым из них может начинаться до того как написана первая строчка программного кода. Руководство пользователя описывает взаимодействие с системой с точки зрения пользователя. Это отличное средство проверки того что концепция команды программистов – разработчиков соответствует мнению клиента. По сколько решение принятые в процессе проработки сценария, соответствуют действиям, которые потребуются от пользователя при использовании программы, то написании руководства пользователя естественным образом увязывается с процессом проработки сценария.

Второй существенный документ – это проектная документация. Она протоколирует основные решения принятые при планировании программы и следовательно должно создаваться в тот момент когда эти решения еще связи в памяти создателей. Обычно осуществляется глобальное описание программной системы в начале разработки, затем совершается переход к уровню отдельных компонент и модулей.

Как руководство пользователя, так и проектная документация уточняются и изменяется в процессе работы в точном соответствии с тем как модифицируется сама программа.

3. Готовность к изменениям

Независимость от того, как тщательно разрабатываются исходные спецификации и проект программной системы почти наверняка изменения в

желаниях или потребностях пользователя будет вызывать соответствующие исправления в программе. Разработчики должны предвидеть это и планировать свои действия соответствующим образом.

1. Главная цель состоит в том что изменения должны затрагивать как можно меньше компонент даже принципиальные новшества во внешнем виде или функционирования приложения должны затронуть один или два фрагмента кода.

2. Необходимо предсказать наиболее вероятные источники изменений и позволить им влиять на возможно меньшее количество компонент программы. Наиболее общими причинами изменения является интерфейс формата обмена информацией вид выходных данных.

3. Необходимо изолировать и уменьшить зависимость программного обеспечения от аппаратуры.

4. Уменьшение количества связей между фрагментами программы снижает взаимосвязь между ними и увеличивает вероятность того что каждую компоненту удастся изменить с минимальным воздействием на другие.

5. Необходимо аккуратно заносить записи о процессе разработки и о дискуссиях проводившихся вокруг принципиальных решений в проектную документацию. Проектная документация позволит в последствии узнать о мотивах принятых решений и поможет избежать затраты времени на обсуждение вопросов которые уже были разрешены.

4. Реализация компонент

Когда проект в целом определен и разбит, на подсистемы следующим шагом является реализация компонент. Если предыдущие этапы были выполнены корректно каждая обязанность или поведение будут кратко охарактеризованы.

Задачей данного этапа является воплощение желаемых действий на компьютерном языке. То есть на данном этапе осуществляется реализация конкретных программ. Важной частью анализа и кодирования является полная характеристика и документирование необходимых предварительных условий, которые требуется программной компоненте, для выполнения задания. Также следует проверить правильно ли работает программная компонента, если ее вызвать ее с правильными входными значениями. Это подтвердит корректность алгоритмов использованных при реализации компонентов.

1. Интеграция компонент

Когда индивидуальные подсистемы разработаны и протестированы, они должны быть интегрированы в конечный продукт. Это делается, поэтапно начиная с элементарной основы к системе, постепенно добавляются новые элементы. При этом для еще не реализованных частей используется так называемые заглушки. Это программы, без какого либо поведения или с ограниченной функциональностью. Заглушки должны выдавать информационные соглашения и возвращать управление.

Отладку отдельных компонент часто называют тестированием блоков. Затем заглушки заменяются более серьезным кодом. Тестирование до тех пор, пока не станет ясно, что система работает правильно. Этот процесс называют

тестированием системы в целом. Во время интеграции системы возможно, что ошибка проявляющаяся в одной из программных систем вызвана не корректным кодом в другом фрагменте, тем самым ошибки выявляемые в процессе необходимости исправлять некоторые компоненты. В след за этим измененные компоненты должны вновь тестироваться изолированно перед очередной попыткой интеграции. Повторная прогонка разработанных ранее тестовых примеров выполняемая после изменений в компоненте программы называется регрессионным тестированием.

1. Сопровождение и развитие

После передачи пользователю функционирующего приложения необходимо дополнительное сопровождение программного обеспечения.

Рассмотрим некоторые причины вызывающие ее неизбежность:

1. В переданном продукте может быть обнаружены ошибки. Они должны быть исправлены либо через «закладки» к существующей версии либо в новой версии.
2. Изменение к требованиям, возможно из за новых государственных постановлений и стандартизации.
3. Переход на другое аппаратное обеспечение.
4. Изменение запросов пользователей. Пользователи могут требовать увеличения возможностей программы более простого использования.
5. Потребность в улучшенной документации.