ПРОГРАММА ПО КУРСУ
«ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С++» (130 Ч)

***Пояснительная записка*:** курс предназначен для тех, кто хочет освоить основы объектно-ориентированного программирования на языке С++ и имеет базовую подготовку. Программа курса предусматривает изучение всех этапов подготовки и написания программ, основных понятий и конструкций языка языке С++. На протяжении всего курса слушатели создадут множество небольших программ, закрепляя полученные теоретические знания на практических примерах. Заключительная часть курса посвящена проектной работе. Учащиеся самостоятельно разработают прикладную программу на основе изученного учебного материала.

СОДЕРЖАНИЕ УЧЕБНОГО ПРЕДМЕТА

Раздел 1. Основы структурного программирования
Языки программирования и компиляторы. Ввод-вывод и переменные. Присваивание, арифметические действия. Использование строк. Алгоритмы и блок схемы. Логические операции и ветвления. Циклы. Работа с файлами.

Раздел 2. Принципы и практика построения алгоритмов
Применение циклов, массивы. Алгоритмы заполнения, поиска, сортировки. Оценка сложности, вложенные циклы. Функции. Структуры. Рекурсия.

Раздел 3. Объектно-ориентированное программирование
Объекты и классы. Инкапсуляция. Наследование. Полиморфизм. Библиотеки классов. Визуальные компоненты и приложения с графическим интерфейсом.

Раздел 4. Разработка проектов
Использование библиотечных классов. Проектирование визуального интерфейса. Иерархия классов. Обработка событий. Структура и организация проекта. Этапы разработки проекта. Курсовая работа.

ТРЕБОВАНИЯ К ПОДГОТОВКЕ УЧАЩИХСЯ

Тема 1. Языки программирования и компиляторы.
Учащиеся должны знать:
- основные понятия: язык программирования, исходный код, машинный код
- понятие компилятор и исполняемый файл
- назначение интерпретируемых языков программирования и сферу их применения

Учащиеся должны уметь:
- установить компилятор и среду разработки
- создать и скомпилировать проект

Тема 2. Ввод-вывод и переменные
Учащиеся должны знать:
- что такое командная строка
- структуру минимального консольного приложения
- понятия операция, оператор и выражение
- понятие поток вывода и поток ввода
- операторы cin и cout
- назначение переменных
- правила именования переменных

- простые типы данных
- понятие инициализация
- что такое литералы и правила их использования

Учащиеся должны уметь:

- выводить информацию на экран в требуемом виде
- выбирать подходящий тип данных для переменной
- описывать и инициализировать переменные
- обрабатывать информацию из потока ввода

Тема 3. Присваивание, арифметические действия

Учащиеся должны знать:

- назначение оператора присваивания
- основные арифметические операторы
- приоритет операций при вычислении

Учащиеся должны уметь:

- присвоить переменной требуемое значение
- составить корректное математическое выражение
- написать программу, выполняющую расчет по представленной формуле

Тема 4. Строки

Учащиеся должны знать:

- класс строк из стандартной библиотеки
- основные функции для работы со строками
- функции преобразования числа в строку и обратно

Учащиеся должны уметь:

- объединить строки в одну
- узнать длину строки
- выделить подстроку в отдельную переменную
- написать программу, составляющую предложение из отдельных слов

Тема 5. Алгоритмы и блок схемы.

Учащиеся должны знать:

- суть структурного подхода к программированию
- понятие алгоритм, его назначение
- виды алгоритмов: линейный, ветвление, циклический
- виды циклов
- понятие блок-схемы, основные блоки

Учащиеся должны уметь:

- составить чёткую и достаточно детализированную последовательность действий
- изобразить алгоритм в виде блок схемы
- понять выполняему алгоритмом задачу по представленной блок-схеме
- различать циклы
- выбирать наиболее подходящий для конкретной задачи вид цикла

Тема 6. Логические операции и ветвления.

Учащиеся должны знать:

- назначение логического типа данных и его возможные значения
- операции сравнения
- логические операции
- оператор ветвления

Учащиеся должны уметь:

- сравнивать значения между собой
- составлять сложные логические выражения
- реализовать алгоритм с ветвлением

Тема 7. Циклы.
Учащиеся должны знать:
- сферу применения циклов и решаемые ими задачи
- понятия тело цикла, условие выхода из цикла
- синтаксис циклов с предусловием, постусловием и параметром

Учащиеся должны уметь:
- выбрать подходящий к задаче вид цикла
- подготовить переменные к началу цикла
- сформулировать условие выхода из цикла
- реализовать тело цикла
- проверить достигает ли цикл условия выхода при всех возможных вариантах входящих данных
- реализовать алгоритм с циклической составляющей

Тема 8. Файлы.
Учащиеся должны знать:
- способы работы с файлами: с использованием файловых переменных или с помощью потоков
- библиотеки содержащие функции для работы с файлами
- классы файловых потоков ifstream и ofstream
- функции файловых потоков и типичные способы их применения

Учащиеся должны уметь:
- описать в программе поток ввода в файл или чтения из файла
- открыть файл с обработкой ошибки в случае неудачи
- организовать циклическое чтение из файла или циклическую запись в файл

Тема 9. Массивы.
Учащиеся должны знать:
- назначение массивов и сферу их применения
- понятия элемент массива, длина массива, тип данных элементов массива
- синтаксис описания и инициализации элементов массива

Учащиеся должны уметь:
- описать и инициализировать массив
- обратиться к элементу массива по индексу
- использовать элементы массива в выражении
- поменять значения элементов массива местами

Тема 10. Циклическая обработка массивов.
Учащиеся должны знать:
- реализацию типичных алгоритмов работы с массивом: вывод всех элементов на экран, заполнение массива значениями, вычисление суммы элементов массива

Учащиеся должны уметь:
- реализовать цикл для чтения и для записи элементов массива
- вычислить сумму, среднее значение элементов массива
- выяснить, имеется ли в массиве элемент с заданным значением
- посчитать элементы массива, удовлетворяющие заданному условию
- найти максимальный или минимальный элемент массива

- реализовать алгоритм бинарного поиска
- реализовать алгоритмы сортировки: гномью, пузырьковую, выбором

Тема 10. Сложность алгоритмов.
Учащиеся должны знать:
- понятие сложность алгоритма
- принцип алгоритма бинарного поиска и быстрой сортировки

Учащиеся должны уметь:
- оценить количество действий в алгоритме, соотнести его с количеством обрабатываемых данных и дать качественную оценку сложности алгоритма
- уметь оценить сложность для лучшего, худшего случая и среднюю сложность

Тема 12. Функции.
Учащиеся должны знать:
- понятие подпрограммы, функции
- цели и преимущества выделения части исходного кода в функции
- синтаксис описания функции
- понятия формальный и фактический параметр функции
- понятие возвращаемое значение, тип возвращаемого значения

Учащиеся должны уметь:
- оценить обоснованность выделения части алгоритма в функцию
- реализовать функцию: определить тип возвращаемого значения, состав и тип данных параметров функции, реализовать тело функции
- использовать функции в выражениях
- реализовать заданный алгоритм с использованием функций

Тема 13. Структуры.
Учащиеся должны знать:
- пользовательские типы данных, преимущества их использования
- синтаксис описания структуры
- как объявить и использовать структуру

Учащиеся должны уметь:
- создать собственную структуру исходя из потребностей задачи
- определить ситуацию, в которой структура упростит достижение поставленной цели или улучшит читаемость исходного кода

Тема 14. Рекурсия.
Учащиеся должны знать:
- принцип построения рекурсивного алгоритма
- понятия база и шаг рекурсии
- примеры простейших рекурсивных алгоритмов
- случаи, когда использование рекурсии целесообразно

Учащиеся должны уметь:
- определить базу и шаг для конкретной задачи
- составить рекурсивный алгоритм
- реализовать рекурсию на языке программирования
- оценить глубину рекурсии и практическую применимость алгоритма
- преобразовать простой рекурсивный алгоритм в итеративный

Тема 15. Объекты и классы.
Учащиеся должны знать:
- понятия класс, объект

- принципы объектно-ориентированного программирования
- понятия поле, свойство, метод класса
- синтаксис описания класса
- модификаторы доступа private и public, их назначение
- понятия конструктор и деструктор, их назначение
- как описать и использовать объект и его методы

Учащиеся должны уметь:

- определить состав полей и методов, которые должны входить в класс
- разбить данные класса на внутренние и внешние
- описать и реализовать функцию-метод класса
- добавить конструктор и деструктор
- создать объект заданного класса
- использовать его функционал

Тема 16. Принципы ООП.

Учащиеся должны знать:

- принцип инкапсуляции, причины его возникновения, почему он так важен
- суть принципа наследования, понятия родительский и дочерний класс
- как строится иерархия классов
- что такое полиморфизм и где он может быть полезен

Учащиеся должны уметь:

- описать производный класс от базового, т.е. выполнить наследование
- привести пример полиморфизма

Тема 17. Библиотеки классов.

Учащиеся должны знать:

- названия и назначение распространенных библиотек классов
- назначение, преимущества и недостатки библиотеки wxWidgets
- способы получения справочной информации по интересующей библиотеке
- возможности среды разработки по использованию библиотеки
- общую структуру иерархии классов wxWidgets

Учащиеся должны уметь:

- выяснить и назвать плюсы, минусы и сферу применения заданной библиотеки классов
- найти справочную информацию по заданной библиотеке и нужному компоненту

Тема 18. Визуальные компоненты.

Учащиеся должны знать:

- средства среды разработки по визуальному проектированию интерфейса
- событийно-ориентированный подход к организации функционирования программы
- понятия события и свойства
- понятие обработчик события
- класс Application и класс Frame
- классы wxStaticText, wxTextCtrl, wxButton

Учащиеся должны уметь:

- создать приложение с использованием визуальных компонентов
- определить события, которые требуется обрабатывать и создать обработчики событий
- создать простое приложение по поставленной задаче, реагирующее на нажатия мышью

Тема 19. Графический интерфейс.

Учащиеся должны знать:

- назначение основных стандартных визуальных компонентов, используемых для формирования графического интерфейса
- возможности взаимодействия компонентов между собой
- невизуальные компоненты, расширяющие возможности программы, такие как wxTimer

Учащиеся должны уметь:

- спроектировать графический интерфейс с такими элементами как поля ввода, многострочный ввод, таблица значений, группа флажков и т.д.
- создавать обработчики событий, управляющие свойствами визуальных компонентов, и использующие информацию из них

Тема 20. Практика по созданию приложений с графическим интерфейсом

Учащиеся должны знать:

- основные принципы и подходы к созданию сложного проекта
- класс wxEvent
- создание общего обработчика событий для группы компонентов

Учащиеся должны уметь:

- выделить набор требуемых компонентов
- спроектировать интерфейс
- установить логические взаимосвязи между компонентами
- спроектировать логику и последовательность наступления и обработки событий
- реализовать на практике работающее приложение согласно поставленной задаче

Тема 21. Структура и организация проекта.

Учащиеся должны знать*:*

- содержание заголовочных файлов проекта
- внесение изменений в описание класса фрейма
- изменение конструктора фрейма
- подключение дополнительных библиотек
- указатель this

Учащиеся должны уметь*:*

- описать приватные поля класса фрейма
- организовать передачу данных между обработчиками событий
- включить поддержку стандартов C++11, 14

Тема 22. Этапы разработки проекта.

Учащиеся должны знать:

- современные подходы к разработке проектов и организация их жизненного цикла
- назначение и польза технического задания
- этапы развития проекта

Учащиеся должны уметь*:*

- сформулировать идею и поставить цель проекта
- сформировать концепцию, макет графического интерфейса, описать список функций проекта
- создать прототип приложения

Тема 23. Курсовая работа.

Учащиеся должны знать:

- как системно и комплексно подойти к разработке индивидуального проекта

Учащиеся должны уметь:

- найти оригинальную идею для личного проекта
- оценить сложность и определить цель проекта

- определить последовательность выполнения этапов разработки и их сроки
- самостоятельно разработать интерфейс
- самостоятельно реализовать содержательную часть проекта
- самостоятельно провести тестирование и исправление ошибок
- провести доработку и внедрение дополнительного функционала

УЧЕБНО–ТЕМАТИЧЕСКИЙ ПЛАН ПО КУРСУ «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С++»

1. Основы структурного программирования

№	Темы	Кол-во занятий	Кол-во часов
1.	Языки программирования и компиляторы.	1	2
2.	Ввод-вывод и переменные.	3	6
3.	Присваивание, арифметические действия.	2	4
4.	Строки.	1	2
5.	Алгоритмы и блок схемы.	2	4
6.	Логические операции и ветвления.	2	4
7.	Циклы.	3	6
8.	Файлы.	1	2
	Всего:	**15**	**30**

2. *Принципы и практика построения алгоритмов*

№	Темы	Кол-во занятий	Кол-во часов
1.	Массивы	2	4
2.	Циклическая обработка массивов.	4	8
3.	Сложность алгоритмов	2	4
4.	Функции	4	8
5.	Структуры	2	4
6.	Рекурсия	2,5	5
7.	**Экзамен за 1 семестр**	1	**2**
	Всего:	**17,5**	**35**
Всего за 1 семестр		**32,5**	**65 ч**

3. Объектно-ориентированное программирование

№	Темы	Кол-во занятий	Кол-во часов
1.	Объекты и классы.	2	4
2.	Принципы ООП.	2	4
3.	Библиотеки классов.	3	6
4.	Визуальные компоненты.	3	6
5.	Графический интерфейс.	4	8
	Всего:	**14**	**28**

4. Разработка проектов

№	Темы	Кол-во занятий	Кол-во часов
1.	Практика по созданию приложений с графическим интерфейсом: использование библиотечных классов, проектирование визуального интерфейса, иерархия классов, обработка событий.	11	22
2.	Структура и организация проекта.	2	4
3.	Этапы разработки проекта.	1	2
4.	Курсовая работа.	3,5	7
5.	**Экзамен за 2 семестр. Защита курсовой работы.**	**1**	**2**
	Всего:	**18,5**	**37**
Всего за 2 семестр:		**32,5**	**65 ч**
Итого за весь курс:		**65**	**130 ч**