**Аннотация к курсу: Язык программирования Microsoft Visual C++**

Язык программирования С++ — это универсальный язык с богатым набором операторов и компактным способом записи выражений. Благодаря гибкости, выразительности и компактности своих конструкций С++ завоевал наибольшую популярность в среде профессиональных программистов и широко используется при разработке системных и прикладных программ.

Язык С++ представляет собой удачный компромисс между желанием располагать теми возможностями, которые обычно предоставляют программисту столь понятные и удобные языки высокого уровня, и стремлением эффективно использовать особенности компьютера.

С помощью С++ можно сделать то, что на Паскале сделать невозможно (или почти невозможно), — например, написать фрагмент операционной системы (или новую операционную систему), утилиты и т. п.

Язык С++ подходит как для системных программистов, так и для написания коммерческого программного обеспечения – включая игры, графику и др. видов задач.

|  |  |
| --- | --- |
| Тема | Кол-во часов |
| 1. Введение в язык программирования С++   Исторически язык С++ является развитием языка программирования С. Эволюционный скачок был ознакомлен к языку С новой парадигмы программирования: объектно-ориентированной. Общий порядок создания программы на С++, которую можно запускать прямо из операционной системы, состоит из последовательности шагов:   1. Создание исходного кода программы (можно в блокноте, но с расширением .cpp. 2. Создание объектного кода программы (текстовый файл передается компилятору с созданием объектного файла с расширением .obj). 3. Создание исполняемого кода программы (результат – исполняемый код). | 1 |
| 1. Элементы С++: алфавит, идентификаторы, литералы, служебные слова   Программа оформляется как отдельный *проект (project)*.  **Проект** (project) представляет собой набор файлов, которые используются для создания программы.  Проекты размещаются в ***рабочих областях*** (workspaces).  **Алфавит** – включает из следующие символы: прописные и строчные латинские буквы и знак подчеркивания; цифры от 0 до 9. Специальных символов + - \* / = > < . ; , : [ ] { } ( ) # $ \ ! % ^ “ & ? ‘  **Лексема** – минимальная единица языка, имеющая самостоятельный смысл.  **Выражение** задает правило вычисления некоторого значения.  **Оператор** задает законченное описание некоторого действия.  Из символов алфавита формируются лексемы языка такие как **идентификаторы, ключевые** (зарезервированные) **слова, знаки операций, константы, переменные, разделители** (точка, запятая, скобки, пробельные символы). | 1 |
| 1. Типы данных и операции в языке. Выражения   **Знаки операций** – один или более символов, определяющих действия над операндами. Операции делятся на **унарные**, **бинарные** и **тернарную** по количеству участвующих в них операндов. **Константы**  – это неизменяемые величины. Они могут быть целые, вещественные, символьные и строковые.  Последовательности символов, начинающиеся **с обратной косой черты**, называются **управляющими последовательностями**.  **Тип данных определяет**:  внутреннее представление данных в памяти компьютера;  множество значений, которые могут принимать величины этого типа;  операции и функции, которые можно принимать к величинам этого типа.  **Основные типы данных**  Для описания основных типов определены следующие ключевые слова:  int – целый;  сhar - символьный  wchar\_t – расширенный символьный, 1 символ – два байта (Unicode);  bool – логический, принимает значение true или false;  float – вещественный;  double – вещественный с двойной точностью.  void используется для определения функций, которые не возвращают значения, для указания пустого списка аргументов функции, как базовый тип для указателей и в операции приведения типов. | 1 |
| 1. Операторы. Управляющие конструкции языка   **Переменная** – это именованная область памяти, в которой хранятся данные определенного типа. Она имеет **имя** и **значение**. Значение переменной может изменяться во время выполнения программы. Перед использованием любая переменная должна быть объявлена.  int a; float r, s ,f;  Общий вид оператора объявления переменных:  [const] тип имя [инициализатор];  Модификатор const  *Инициализация* – присвоение начального значения при описании переменной.  Программу можно составить из трех *базовых конструкций*:   1. **следование;** 2. **ветвление;** 3. **циклы.**   **Поток** – это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику. Чтение данных из потока – ***извлечение***, вывод данных в поток – ***включение***.  Потоки подразделяются на:   * *входные потоки* – данные вводятся в память; * *выходные потоки* – данные выводятся из памяти; * *двунаправленные* – возможно как включение, так и извлечение. * *стандартные* потоки – ввод с клавиатуры и вывод на экран. | 1 |
| 1. Линейные программы. Оператор присваивания   Программа состоит из **функций**, **описаний** и **директив препроцессора**. **Функция** - именованный набор описаний и операторов (инструкций) сгруппированных для решения определенной задачи. Функция может принимать параметры и возвращать значение. Одна из функций должна иметь имя **main**. Выполнение программы начинается с первого оператора этой функции.  Структура функции:  **тип\_возвращаемого\_значения имя ([параметры])**  **{**  **тело функции**  **}**   * если функция не должна возвращать значение, указывается тип **void**; * тело функции является блоком, и, следовательно, заключается в фигурные скобки; * функции не могут быть вложенными; * каждый оператор заканчивается точкой с запятой, кроме составного оператора.   *Оператор* представляетсобой выражениезаканчивающееся точкой с запятой.  Примеры:  i++;  a\*=b + c; // Присваивание с умножением  fun(f, k);  z=5\*x+x\*y; // Присваивание | 2 |
| 1. Разветвляющие программы. Оператор IF   ***Условный оператор if***  используется для разветвления процесса вычислений на два направления. Формат оператора:  **if ( <условие> ) <оператор 1 >; else <оператор 2>;**  Одна из ветвей может отсутствовать.  *Составной оператор* – это несколько операторов, объединенных в фигурные скобки.  ***Условный оператор if***  может используется для разветвления процесса вычислений более чем на два направления. Формат оператора:  **if ( <условие 1> ) <оператор 1 >;**  **else if (условие 2 ) <оператор 2>; else <оператор 3>;** | 4 |
| 1. Разветвляющие программы. Оператор выбора (переключатель) SWITCH   ***Оператор switch*** (переключатель) используется для разветвления процесса вычислений на несколько направлений. Формат оператора:  **switch ( <выражение или переменная> ) {**  **case <**constant1>: <последовательность операторов> ; **break;**  **case <**constant2>: <последовательность операторов> **; break;**  **…**  **case <**constant N>: <последовательность операторов> **; break;**  **default : <**последовательность операторов>**;**  **}** | 4 |
| 1. Циклические программы. Оператор цикла FOR   *Циклы*  Циклы применяются для организации многократного повторения вычислений.  Цикл состоит из:   1. ***заголовка цикла*** 2. ***тела цикла*** - операторов (инструкций), которые выполняются несколько раз; 3. начальных установок параметров цикла; 4. модификации ***параметров*** цикла и проверки условия продолжения выполнения цикла.   ***Итерация*** – один проход цикла.  Проверка условия выполняется на каждой итерации либо до тела цикла (*цикл с предусловием*), либо после тела цикла (*цикл с постусловием*).  Параметры цикла -переменные, изменяющиеся в теле цикла и используемые при проверке условия продолжения.  Цикл с параметрами (**for**) – цикл с предусловием  Формат оператора цикла:  **for ( <инициализация>; <проверка условия>; <изменение> )**  **Тело цикла**  **Инициализация** – присвоение начального значения параметру (переменной) цикла.  **Проверка условия** - условное выражение, которое определяет, когда цикл должен быть завершен. Если условие истинно , выполняются операторы, составляющие тело цикла.  **Изменение** (модификация) выполняется после каждой итерации цикла и служит для изменения параметров (переменной) цикла. Можно записать несколько параметров через запятую.  Любая из частей оператора for может быть опущена(но точки с запятой надо оставлять на своих местах). | 4 |
| 1. Циклические программы. Оператор цикла WHILE   Цикл с предусловием (**while**)  Формат оператора цикла:  **while (<условие>)**  **Тело цикла;**  **Тело цикла**  может состоять из одного или нескольких операторов. Если тело цикла > 1 оператора, то тело цикла заключается в фигурные скобки.  **Условие** - логическое выражение, которое проверяется для выхода из цикла. Цикл выполняется до тех пор, пока условие истинно. Когда условие ложно, программа передает управление следующему по порядку оператору. | 4 |
| 1. Циклические программы. Оператор цикла DO … WHILE   Цикл с постусловием (**do while**)  Формат оператора цикла:  **do**  **Тело цикла**  **while (<условие>);**   1. Выполняются операторы, составляющие тело цикла. 2. Проверяется условие. Если условие истинно – выполнение цикла повторяется | 4 |
| 1. Вложенные циклы. Массивы   **Массив** - именованная область памяти состоящая из конечного набора однотипных элементов, каждый элемент имеет свой порядковый номер. Порядковый номер называется индексом элемента в массиве. Элементы массива нумеруются с нуля.  Общий вид оператора описания массива:  **тип имя [размерность][инициализатор]**;  *Инициализация* – присвоение начальных значений при описании массива. Инициализатор (необязательный). Инициализируемые значения записываются в фигурных скобках – со знаком равенства . Если элементов в массиве больше, чем инициализаторов, то остальные элементы обнуляются.  Для доступа к элементу массива после его имени указывается индекс в квадратных скобках.  Одномерный массив – вектор.  *Инициализация* **м**ногомерныхмассивов представляется в виде:   * + массива из массивов, при этом каждый массив заключается в свои фигурные скобки;   + общего списка элементов в том порядке, в котором элементы располагаются в памяти.   int mass1 [][2]= {{3,2}, {2,5}, {-2,1}}; | 4 |
| 1. Функции   **Функция** – это последовательность описаний и операторов, сгруппированных для решения конкретной задачи.  **Функция имеет собственное имя.**  *Определение функций*  **тип имя ([список параметров])**  **{тело функции}**  В тексте программы либо саму **функцию**, либо ее **прототип** располагают до главной функции **main**.  *Способы передачи параметров в функцию*:   1. *Передача параметров по значению* 2. *Передача по адресу* 3. *Передача по ссылке* | 4 |
| 1. Обработка символьной информации   ***Строка*** - массив символов, заканчивающийся нуль-символом. ***Нуль-символ*** – это символ с кодом, равным 0, записывается в виде управляющей последовательности ‘\0’. По положению нуль-символа определяется фактическая длина строки.  **char str [10] =“Vesta”; //**массив из 10 элементов.  // str[0]=‘V’ , str[1]=‘e’, str[2]=‘s’, str[3]=‘t’, str[4]=‘a’, str[5]=‘\0’  Если строка при определении инициализируется, то размерность можно не задавать.  **Ввод-вывод строк –** используются объекты ***cin***, ***cout.*** | 2 |
| 1. Файлы (Потоки)   **Поток** – это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику. Чтение данных из потока – ***извлечение***, вывод данных в поток – ***включение***.  Потоки подразделяются на:   * *входные потоки* – данные вводятся в память; * *выходные потоки* – данные выводятся из памяти; * *двунаправленные* – возможно как включение, так и извлечение.   По виду устройств с которыми работает поток подразделяются на:   * *стандартные* потоки – ввод с клавиатуры и вывод на экран; * *файловые* потоки – обмен информацией с файлами на внешних носителях; * *строковые* потоки – работа с массивами символов в оперативной памяти.   Файловые потоки  <fstream.h> - заголовочный файл для работы с файлами.  <istream.h> - заголовочный файл для входных потоков.  <ostream.h> - заголовочный файл для выходных потоков. | 2 |
| 1. Обобщение. Решение задач | 2 |