Министерство образования и науки Мурманской области
Государственное автономное учреждение дополнительного образования
Мурманской области
«Мурманский областной центр дополнительного образования
«Лапландия»

ПРИНЯТА
методическим советом
Протокол
от 06.09.2017 № 1

Председатель _____ О. А. Бережняк

УТВЕРЖДЕНА
Приказом ГАУДО МО «МОЦДО
«Лапландия»
от 07.09.2017 № 521

Директор _____ С. В. Кулаков

ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ
ОБЩЕРАЗВИВАЮЩАЯ ПРОГРАММА
ТЕХНИЧЕСКОЙ НАПРАВЛЕННОСТИ

«ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ»

Возраст учащихся: **13-17 лет**
Срок реализации программы: **2 года**

Автор:
Павлов Николай Александрович,
педагог дополнительного образования

Мурманск
2017

Пояснительная записка

Стремительное развитие средств вычислительной техники во второй половине XX века привело к созданию концепции персонального компьютера. Электронная вычислительная машина, ранее доступная только немногим специалистам, в начале XXI века стала доступна любому человеку.

Не менее стремительно шло развитие средств (языков) и концепций программирования. На смену языкам низкого уровня, сильно привязанным к аппаратным средствам, приходили языки высокого уровня абстракции, позволявшие составлять программы, практически не беспокоясь о том, на каком именно компьютере они будут выполняться. Концепция линейной программы, составленной из множества следующих друг за другом операторов, была заменена сначала концепцией процедурного программирования, а затем – концепцией объектно-ориентированного программирования (ООП). Последняя является доминирующей концепцией во всех ныне существующих языках программирования высокого уровня и, наиболее вероятно, будет оставаться доминирующей в течение следующих 5-10 лет.

Актуальность программы

На сегодняшний день для развития программного обеспечения крайне необходимо использовать современные концепции программирования, которые позволяют создавать конкурентоспособное программное обеспечение.

Изучение объектно-ориентированного программирования даёт возможность обучающимся использовать мощный инструмент для разработки программного обеспечения и позволяет в новом аспекте рассматривать реальный мир, как систему, выделяя схожие компоненты и понимая принципы их взаимодействия.

Освоение принципов объектно-ориентированного программирования помогает в решении как узких инженерных задач, стоящих перед разработчиками программного обеспечения, так и широких естественнонаучных проблем, позволяя строить эффективные модели реального мира, что согласуется с основными направлениями развития дополнительного образования, изложенными в «Концепции развития дополнительного образования детей», утвержденной распоряжением №1726-р, подписанным Д. Медведевым 4 сентября2014г.

Программа составлена на основе:

- дополнительной образовательной программы «программа «Современное программирование» (авторы А.С. Матвеев, Р.А. Кабетов, Ф.А. Вомпе, Ю.А. Пюрбеев).
- специальной литературы по данному виду технического творчества;
- профессионального опыта педагога.

Отличия

Дополнительная общеобразовательная программа «Современное программирование»	Дополнительная общеобразовательная программа «Основы объектно-ориентированного программирования»
Изменен возрастной диапазон	
возраст учащихся 12-17 лет	возраст учащихся 13-17 лет
Определена цель программы, исходя из поставленной цели сформулированы новые задачи и ожидаемые результаты	
научить современным принципам программирования, сформировав навыки разработки программного обеспечения, обеспечивающие дальнейшую профориентацию обучающихся.	удовлетворение образовательных потребностей учащихся в области программирования и разработки приложений с графическим интерфейсом на базе современных сред разработки программного обеспечения.
Изменен учебно-тематический план	
	См. «Учебно-тематический план»
Содержание программы	
	Убран раздел касающийся изучения «Веб-программирование» Изменено содержание программы в связи с изменением базовой среды разработки (вместо Qt используется Visual Studio)
Добавлена система оценки и фиксирования образовательных результатов	
Отсутствует	См. «Система оценки и фиксирования образовательных результатов

Вид программы: модифицированная.

Направленность программы: техническая.

Срок реализации программы 2 года.

Программа рассчитана на 144 часа.

Режим занятий: 2 раза в неделю по 2 часа.

Режим занятий соответствует санитарно–эпидемиологическим требованиям к учреждениям дополнительного образования детей (санитарно- эпидемиологическим правилам и нормативам СаНПин 2.4.4 3172- 14).

Форма организации занятий: групповая.

Возраст учащихся:13-17 лет.

Количество учащихся: до 15 человек.

Состав группы: постоянный.

Набор учащихся: свободный.

Учащиеся зачисляются в учебные группы при отсутствии медицинских противопоказаний. Обязательно наличие медицинской справки о состоянии здоровья и допуске к занятиям в объединении.

В дополнительной общеобразовательной программе предусмотрен **летний блок занятий** по индивидуальным планам учащихся на летний период времени.

Цель программы

Удовлетворение образовательных потребностей учащихся в области программирования и разработки приложений с графическим интерфейсом на базе современных сред разработки программного обеспечения.

Основные задачи

Обучающие

– ознакомить с принципами функционирования и основными узлами современного персонального компьютера;

– дать знания о базовых понятиях программирования;

– ознакомить с принципами создания программ под платформу Windows;

– дать знания по интегрированным средам разработки (Visual Studio);

– ознакомить с разнообразием методов решения инженерно-технических задач.

Развивающие

– развить процедурный и объектно-ориентированный подход к разработке программного обеспечения на примере языка C++;

– сформировать стиль программирования;

– развить умение искать ошибки и тестировать программы;

– развить умение работать с литературой и справочными файлами, ориентироваться в информационном пространстве, анализировать, обобщать, делать выводы;

– повысить уровень эффективности создаваемого кода;

– развить речевую деятельность посредством выступлений с отчётами на конференциях;

– повысить уровень образного и абстрактного мышления;

– развить умения работать индивидуально и в команде.

Воспитательные

- воспитывать аккуратность, трудолюбие, целеустремленность,
- воспитывать адекватное отношение к личным творческим успехам и успехам других.

Ожидаемые результаты обучения:

Знать

- основы объектно-ориентированного программирования;
- среду разработки Visual Studio;
- синтаксис и семантику языка программирования C++

Уметь

- уметь применять процедурный и объектно-ориентированный подход к разработке программного обеспечения на примере языка C++;

- ориентироваться в информационном пространстве, анализировать, обобщать, делать выводы;
- уметь искать ошибки и тестировать программы;
- уметь повысить уровень эффективности создаваемого кода.
- работать в среде разработки Visual Studio;
- разрабатывать приложения с графическим интерфейсом средствами среды разработки Visual Studio.

Формы диагностики результатов обучения: наблюдение, проверочные задания, тесты, самостоятельные практические работы, соревнования, конференции, семинары.

II. Учебно-тематический план первого года обучения

№	Тема	Количество часов		
		Теория	**Практика**	**Всего часов**
1	**Введение**	**2**	**-**	**2**
1.1	Собеседование	1	-	1
1.2	Вводное занятие. Техника безопасности	1	-	1
2	**Алгоритмы**	**13**	**15**	**28**
2.1	Устройство компьютера. Алгоритмы. Блок-схемы	2	2	4
2.2	Элементарные алгоритмы	-	2	2
2.3	Понятие переменной, типа данных, оператора. Структура программы в С++	2	-	2
2.4	Ввод-вывод в текстовом режиме	1	3	4
2.5	Арифметические операции. Тип *double*	2	2	4
2.6	Базовый синтаксис С++	-	2	2
2.7	Циклы	4	-	4
2.8	Решение задач с использованием циклов обоих типов	-	4	4
2.9	Сравнение языка С++ с другими языками ООП.	2	-	2
3	**Обработка массивов. Функции**	**14**	**22**	**36**
3.1	Массивы. Заполнение массивов. Вывод содержимого массива на экран	4	8	12
3.2	Логический тип данных. Логические операции	2	4	6
3.3	Сортировка массива (выборка, пузырёк)	2	6	8
3.4	Функции. Передача параметров по ссылке и по значению	4	-	4
3.5	Матрицы	2	4	6
4	**Строки и символы. Файлы**	**8**	**20**	**28**
4.1	Символьный тип данных	2	4	6
4.2	Строки в стиле *С*	2	4	6
4.3	Файлы	2	4	6
4.4	Обработка текста	2	8	10
5	**Графика**	**12**	**8**	**20**
5.1	Графика: точка, линия, прямоугольник, закрашивание областей	4	-	4
5.2	Линейное движение объектов	4	4	8
5.3	Вращение объектов	4	4	8
6	**Основы объектно-ориентированного программирования (ООП)**	**7**	**17**	**24**
6.1	Структуры. Общее понятие объекта	1	1	2
6.2	Классы. Спецификаторы доступа	1	1	2
6.3	Базовые принципы объектно-ориентированного моделирования	2	2	4
6.4	Разработка итоговых проектов	3	13	16
7	**Итоговое занятие**	**4**	**2**	**6**
	Итого	**60**	**84**	**144**

Учебно-тематический план второго года обучения

№	Тема	Количество часов		
		Теория	Практика	Всего часов
1	**Введение**	**6**	**2**	**8**
1.1	Вводное занятие. Техника безопасности	1	-	1
1.2	Обзор задач курса	1	-	1
1.3	Графика под Win32	2	-	2
1.4	Повторение: указатели	2	2	4
2	**Повторение: Объектно-ориентированный подход и его реализация в языке C++**	**10**	**14**	**24**
2.1	Понятие класса как типа данных	2	-	2
2.2	Инкапсуляция. Модификаторы доступа. Задача моделирования движения шаров	1	3	4
2.3	Моделирование столкновения шаров	2	2	4
2.4	Перегрузка операций	1	3	4
2.5	Наследование. Задача о геометрических фигурах	2	2	4
2.6	Полиморфизм. Виртуальные функции и классы	2	4	6
3	**Специальные возможности языка C++**	**6**	**4**	**10**
3.1	Приведение типов, указателей и ссылок	2	2	4
3.2	Обработка и генерация исключений	2	2	4
3.3	Механизм *RTTI*	2	-	2
4	**Эффективные алгоритмы**	**10**	**12**	**22**
4.1	Простые алгоритмы сортировки массива. Эффективность алгоритмов	2	2	4
4.2	Быстрая сортировка	2	4	6
4.3	Пирамидальная сортировка	2	2	4
4.4	Генерация комбинаторных объектов	2	2	4
4.5	Перебор с возвратом	2	2	4
5	**Программирование под Windows**	**16**	**14**	**30**
5.1	Основы программирования под платформу *Win32*. Базовые принципы. Обзор библиотек классов-обёрток	4	-	4
5.2	Введение в интегрированную среду разработки Visual Studio	2	4	6
5.3	Обзор библиотек Visual Studio среды разработки Visual Studio: принципы построения приложения, иерархия классов, ограничения	2	-	2
5.4	Библиотеки Visual Studio	6	8	14
5.5	Динамически подключаемые библиотеки	2	2	4
6	**Шаблоны в языке C++**	**12**	**12**	**24**
6.1	Принципы программирования с использованием шаблонов. Простейшие шаблоны функций и классов. Основные понятия (инстанцирование, парадигма)	4	4	8
6.2	Параметры шаблонов. Параметры по умолчанию. Специализация и частичная специализация. Вывод параметров	4	4	8
6.3	Специальное использование ключевых слов в шаблонах. Дружественные шаблоны	2	-	2
6.4	Свойства и стратегии. Идентификация типов	2	4	6

7	**Структуры данных, основные алгоритмы и их реализации на языке С++**	**4**	**4**	**8**
7.1	Структуры и способы хранения данных	2	-	2
7.2	Длинные числа, способ их хранения, ввод и вывод, основные арифметические операции над ними (сложение, вычитание, умножение, деление)	2	4	6
8	**Разработка итоговых проектов**	**2**	**10**	**12**
10	**Итоговое занятие**	**-**	**6**	**6**
	Итого	**66**	**78**	**144**

III. Содержание программы

Первый год обучения

Раздел 1. Введение. (2 ч.)

Тема 1.1. Собеседование.

Собеседование с детьми (и родителями) для определения их интересов и уровня знаний. Анкетирование с целью выявления интересов и ожиданий. Входная диагностика.

Тема 1.2. Вводное занятие. Техника безопасности.

Цель, задачи программы. План работы на учебный год. Режим занятий. Вводный инструктаж по ОТ и ПБ. Первичный инструктаж по темам: «Правила поведения в центре «Лапландия», «Охрана жизни и здоровья обучающихся на учебных занятиях».

Раздел 2. Алгоритмы. (28 ч.)

Тема 2.1. Устройство компьютера. Алгоритмы. Блок-схемы.

Общие принципы функционирования ЭВМ, структура компьютера. Необходимость формализации задачи перед её решением на компьютере. Введение понятия алгоритма. Примеры различных способов записи алгоритмов.

Практическая часть. Запись алгоритмов с помощью блок-схем.

Тема 2.2. Элементарные алгоритмы.

Практическая часть. Решение задач на составление алгоритмов.

Тема 2.3. Понятие переменной, типа данных, оператора. Структура программы в С++.

Понятие оперативной памяти, переменной, типа данных. Объявление и инициализация переменных в С++. Понятие целочисленного типа данных int. Понятие «оператор», оператор присваивания, условный оператор. /*Связь между программой на языке С++ и блок-схемой*/

Тема 2.4. Ввод-вывод в текстовом режиме.

Ввод и вывод данных с помощью стандартных потоков ввода-вывода cout и cin. Ввод и вывод данных с помощью стандартных функций printf и scanf.

Практическая часть. Решение задач с помощью cout и cin; решение задач с помощью printf и scanf.

Тема 2.5. Арифметические операции. Тип *double*.

Знакомство с некоторыми возможностями библиотеки math. h.

Практическая часть. Решение задач.

Тема 2.6. Базовый синтаксис С++.

Практическая часть. Решение задач на «Базовый синтаксис С++».

Тема 2.7. Циклы.

Понятие цикла с предусловием и постусловием. Конструкции языка, позволяющие реализовывать данные циклы в программном коде.

Тема 2.8. Решение задач с использованием циклов обоих типов.

Практическая часть. Решение задач с использованием циклов обоих типов.

Тема 2.9. Язык программирования PHP, сравнение с С++.

Язык программирования PHP, сравнение с языком С++. Возможности, основные отличия.

Раздел 3. Обработка массивов. Функции. (36 ч.)

Тема 3.1. Массивы. Заполнение массивов. Вывод содержимого массива на экран.

Синтаксис объявления и инициализации массивов. Простейшие алгоритмы заполнения и вывода содержания массивов. Принципы поэлементного обхода массивов, ввод понятия счётчика.

Практическая часть. Решение задач на массивы.

Тема 3.2. Логический тип данных. Логические операции.

Понятие логического типа и логических операций: И, ИЛИ, НЕ. Использование логических операций для построения сложных условий.

Практическая часть. Решение задач на логические операции.

Тема 3.3. Сортировка массива (выборка, пузырёк).

Принципы работы алгоритмов сортировки выборкой и пузырьком, самостоятельное построение блок-схемы и реализация на их основе программы на C++.

Практическая часть. Решение задач на сортировку.

Тема 3.4. Функции. Передача параметров по ссылке и по значению.

Удобство использования функций (на примере задачи сортировки массивов), понятия параметров и передачи параметров в функцию.

Тема 3.5. Матрицы.

Понятие двумерных массивов, области применения. Использование в C++, способы передачи массивов в функции.

Практическая часть. Решение задач на обработку массивов (закрепление пройденного материала). Решение задач на матрицы.

Раздел 4. Строки и символы. Файлы. (28 ч.)

Тема 4.1. Символьный тип данных.

Понятие символьного типа данных, типа string. Решение задач по обработке строки: поиск подстроки в строке, замена подстроки и др.

Практическая часть. Решение задач на символьный тип данных.

Тема 4.2. Строки в стиле *C*.

Понятие указателя. Принципы обработки строк в стиле C. Отрицательные моменты использования строк в стиле C.

Практическая часть. Решение задач на строки в стиле C.

Тема 4.3. Файлы.

Структура FILE, файловая переменная. Функции работы с файлами: fscanf(), fprintf(). Простые задачи на запись/чтение файлов.

Практическая часть. Запись в файл чисел, вводимых с клавиатуры. Вывод чисел, записанных в файле. Запись в файл чисел, хранящихся в матрице. Чтение чисел из файла в матрицу. Формирование динамического массива слов на основании текстового файла.

Тема 4.4. Обработка текста.

Более сложные задачи по обработке текстовых файлов: поиск в тексте подстроки, замена слов, подсветка в тексте слов из «словаря».

Практическая часть. Решение задач на обработку текста.

Раздел 5. Графика. (20 ч.)

Тема 5.1. Графика: точка, линия, прямоугольник, закрашивание областей.

Графика различного вида: точка, линия, прямоугольник, закрашивание областей.

Тема 5.2. Линейное движение объектов.

Скорость. Ускорение. Равномерное и равнопеременное движение. Представление движения в пространстве в виде суперпозиции движений по координатам. Вычисление координат объекта по текущим координатам.

Практическая часть. Моделирование броска камня под заданным наперёд углом.

Тема 5.3. Вращение объектов.

Вращение объектов. Синус и косинус.

Практическая часть. Моделирование движения объекта по окружности.

Раздел 6. Основы объектно-ориентированного программирования (ООП). (24 ч.)

Тема 6.1. Структуры. Общее понятие объекта.

Формализация задачи для её объектно-ориентированного решения. Принцип инкапсуляции. Построение диаграммы классов. Синтаксис работы со структурами.

Практическая часть. Формализация задач, запись архитектуры системы в виде диаграмм классов.

Тема 6.2. Классы. Спецификаторы доступа.

Классы и структуры. Отличия классов и структур. Спецификаторы public, private.

Практическая часть. Решение задач из темы 6.1., но с использованием классов и спецификаторов доступа.

Тема 6.3. Базовые принципы объектно-ориентированного моделирования.

Инкапсуляция, полиморфизм, наследование. Примеры преимущества ООП, отличия от процедурного подхода.

Практическая часть. Решение задач с использованием наследования и виртуальных функций.

Тема 6.4. Разработка итоговых проектов.

Разработка элементарных игр типа «Змейка», «Стрелок» с использованием элементов объектно-ориентированного подхода. Преимущества ООП перед процедурным подходом на примере разработки этих игр. Различные способы формализации одних и тех же исходных задач.

Практическая часть. Разработка игр.

Раздел 7. Итоговое занятие. (6 ч.)

Обсуждение итогов работы за год. Обсуждение задания на летний период. Представление мероприятий на следующий год. Итоговая диагностика.

Второй год обучения

Раздел 1. Введение. (8 ч.)

Тема 1.1. Вводное занятие. Техника безопасности.

Основные правила и требования техники безопасности и противопожарной безопасности при работе в компьютерном классе. Цели и задачи программы «Современное программирование» 2-го года обучения.

Тема 1.2. Обзор задач курса.

Краткий обзор решаемых в курсе задач. Темы (на выбор) итоговой работы.

Тема 1.3. Графика под Win32.

Библиотеки, позволяющие писать программы с графическим интерфейсом. Подключение и использование HGE.

Тема 1.4. Повторение: указатели.

Понятие указателя. Операции &, *, new, delete. Преобразование типов указателей. Обращение к объекту структуры через указатель.

Практическая часть. Реализация класса «Динамический массив».

Раздел 2. Повторение: Объектно-ориентированный подход и его реализация в языке C++. (24 ч.)

Тема 2.1. Понятие класса как типа данных.

Компонентные функции**.** Напоминание идей объектно-базированного программирования на примере класса TSpisok.

Тема 2.2. Инкапсуляция. Модификаторы доступа. Задача моделирования движения шаров.

Инкапсуляция. Модификаторы доступа public и private. Задача моделирования движения шаров.

Практическая часть. Моделирование движения шаров, используя класс TSpisok, работу с графикой в Win32.

Тема 2.3. Моделирование столкновения шаров.

Понятие вектора, сложение, умножение на число. Векторы в механике: скорость, ускорение, сила. Законы механики: второй закон Ньютона, законы сохранения импульса и энергии.

Практическая часть. Реализация законов механики в предыдущей задаче.

Тема 2.4. Перегрузка операций.

Перегрузка операций на примере реализации класса Tvector.

Практическая часть. Реализация класса Tvector.

Тема 2.5. Наследование. Задача о геометрических фигурах.

Механизм наследования. Использование наследования для классификации объектов и их функциональности. Наследование как эффективный способ повторного использования кодов. Множественное наследование и его особенности в языке *C++*.

Практическая часть. Добавление объектов разного типа в программе моделирования столкновения шаров с использованием механизма наследования. Задача о геометрических фигурах.

Тема 2.6. Полиморфизм. Виртуальные функции и классы.

Виртуальные функции как механизм реализации динамического полиморфизма. Требования к виртуальным функциям и условия сохранения виртуальности. Размер объекта класса, содержащего виртуальные функции. Виртуальные конструкторы и деструкторы, область их применения. Чистые виртуальные функции. Абстрактные классы, их признаки и ограничения на использование. Организация интерфейсов.

Практическая часть. Создание общего класса-интерфейса массива и реализация его возможностей на случаи статического и динамического массивов.

Раздел 3. Специальные возможности языка *C++*. (10 ч.)

Тема 3.1. Приведение типов, указателей и ссылок.

Приведение (преобразование) одного типа к другому. Использование преобразования в стиле *C*. Использование конструкторов и псевдоконструкторов. Перегрузка операций преобразования типов. Использование стандартных ключевых слов *static_cast*, *dynamic_cast*, *const_cast* и *reinterpret_cast*.

Практическая часть. Реализация метода и операции копирования на уровне базового класса.

Тема 3.2. Обработка и генерация исключений.

Методы перехвата и обработки исключительных ситуаций (ошибок). Конструкция *try catch*. Идентификатор исключительной ситуации и его параметры. Генерация исключительной ситуации. Ключевое слово *throw*.

Практическая часть. Решение задач на исключительные ситуации.

Тема 3.3. Механизм *RTTI*.

Механизм идентификации типов во время выполнения программы (*RTTI*). Необходимое условие возможности использования этого механизма. Ключевое слово *typeid*.

Раздел 4. Эффективные алгоритмы. (22 ч.)

Тема 4.1. Простые алгоритмы сортировки массива. Эффективность алгоритмов.

Сортировки пузырьком, простым выбором, вставкой. Оценка эффективности алгоритмов.

Практическая часть. Реализация пройденных алгоритмов.

Тема 4.2. Быстрая сортировка.

Оценка эффективности алгоритма быстрой сортировки в среднем и в самом худшем случае. Рекурсивная реализация и реализация, использующая стек.

Практическая часть. Реализация быстрой сортировки.

Тема 4.3. Пирамидальная сортировка.

Построение сбалансированного дерева. Сортировка построенного дерева.

Практическая часть. Реализация пирамидальной сортировки массива.

Тема 4.4. Генерация комбинаторных объектов.

Сочетания, перестановки. Задача о переборе всех путей. Порождение комбинаторных объектов.

Практическая часть. Решение комбинаторных задач.

Тема 4.5. Перебор с возвратом.

Задача о ферзях. Отличие от простого перебора.

Практическая часть. Задачи на использование алгоритма перебора с возвратом.

Раздел 5. Программирование под Windows. (30 ч.)

Тема 5.1. Основы программирования под платформу *Win32*. Базовые принципы. Обзор библиотек классов-обёрток.

Создание приложений с использованием программных интерфейсов (API). Особенности интерфейса *WinAPI*, его недостатки. Преимущества, недостатки и сложности программирования с использованием API. Применение ООП для создания приложений под платформу *Win32*, классы-обёртки. Обзор существующих библиотек классов-обёрток (*MFC*, *OWL*, *VCL*, *ATL*, *WTL*).

Тема 5.2. Введение в интегрированную среду разработки Visual Studio.

Интерфейс Visual Studio, инструменты (редактор кодов, инспекторы классов, объектов и свойств), палитра компонентов, простейшие компоненты (кнопка и однострочное окно редактирования). Принципы создания приложений.

Практическая часть. Создание на Visual Studio простейшего приложения «Привет мир!».

Тема 5.3. Обзор библиотек среды разработки *Visual Studio*: принципы построения приложения, иерархия классов, ограничения.

Принципы построения библиотеки среды разработки Visual Studio. Иерархия классов и краткий обзор их функциональных возможностей и назначения. Ограничения библиотеки Visual Studio.

Тема 5.4. Библиотеки Visual Studio**.**

Обзор классов библиотеки. Реализация графических интерфейсов.

Практическая часть. Написание простой программы с графическим интерфейсом, основанным на работе с Visual Studio.

Тема 5.5. Динамически подключаемые библиотеки.

Что такое DLL. Статическое и динамическое подключение DLL. Экспорт функций средствами WinAPI. Экспорт классов.

Практическая часть. Подключение и использование модуля libXML. Формирование своего DLL.

Раздел 6. Шаблоны в языке *C*++. (24 ч.)

Тема 6.1. Принципы программирования с использованием шаблонов. Простейшие шаблоны функций и классов. Основные понятия (инстанцирование, парадигма).

Автоматическая генерация кода. Шаблоны как способ создания однотипных функций и классов. Принципы программирования с использованием шаблонов. Синтаксическая конструкция «шаблон». Объявление и определение простейших шаблонов функций и классов. Понятия инстанцирования (генерации кода по шаблону) и парадигмы (ограничений на параметры шаблонов, связанные со структурой этих шаблонов).

Практическая часть. Обобщение класса динамического массива на случай элементов любого типа; создание функции, производящей поэлементное суммирование двух динамических массивов произвольного типа.

Тема 6.2. Параметры шаблонов. Параметры по умолчанию. Специализация и частичная специализация. Вывод параметров.

Параметризация кода. Параметры шаблонов, являющиеся типами, параметры, не являющиеся типами, и шаблонные параметры шаблонов. Параметры шаблонов по умолчанию. Специализация шаблонов как способ эффективной адаптации общего вида кода к конкретному значению параметров. Частичная специализация. Вывод параметров шаблона по параметрам функции. Явное и неявное указание параметров шаблона.

Практическая часть. Реализация функции *GetMin()* или *GetMax()*, возвращающей минимальное (максимальное) число заданного типа; обобщение функции суммирования на случай произвольной поэлементной операции с помощью реализации дополнительных шаблонов-функторов.

Тема 6.3. Специальное использование ключевых слов в шаблонах. Дружественные шаблоны.

Специальное использование ключевых слов *template* и *typename* в шаблонах для исключения неопределённостей и связанные с этим ограничения. Дружественные шаблоны функций и дружественные шаблоны классов, особенности их объявления и определения. Использование синонимов для сокращения кода.

Тема 6.4. Свойства и стратегии. Идентификация типов.

Свойства и стратегии типов. Реализация свойств с помощью конструкторов, шаблонов функций и шаблонов классов. Простейшие свойства (нуль и единица). Параметризация типов параметров функций и типов возвращаемых значений. Стратегия продвижения. Использование шаблонов для идентификации и классификации произвольных типов.

Практическая часть. Создание класса свойств чисел, содержащего параметры типа: нуль, единица, минимальное и максимальное значение; реализация функции, которая возвращает *true,* если её параметр является целым числом и *false* в противном случае.

Раздел 7. Структуры данных, основные алгоритмы и их реализации на языке *C*++. (8 ч.)

Тема 7.1. Структуры и способы хранения данных.

Основные структуры хранения данных. Линейный список и его частные случаи: стек, очередь, двусторонняя очередь. Операции над линейными списками. Способы хранения линейного списка в памяти ЭВМ. Последовательное и связанное распределение. Разреженная матрица. Деревья и бинарные деревья. Дерево поиска. Удаление элемента из дерева.

Тема 7.2. Длинные числа, способ их хранения, ввод и вывод, основные арифметические операции над ними (сложение, вычитание, умножение, деление).

Проблема хранения больших чисел в памяти ЭВМ и выполнения операций над ними. Понятие длинного числа. Способы хранения длинных чисел в памяти ЭВМ. Выполнение над длинными числами базовых арифметических операций (сложение, вычитание, умножение, деление) и операций ввода-вывода.

Практическая часть. Создание класса длинного числа, реализующего один из способов его хранения. Перегрузка для этого класса базовых операций. Реализация функций преобразования от обычных чисел к длинным числам, а также функций ввода-вывода (преобразования к символьной строке и наоборот). С помощью класса длинного числа реализация вычисления числа 100!.

Раздел 8. Разработка итоговых проектов. (12 ч.)

Выполнение итогового проекта: планирование, реализация и защита программы. Использование своих реализаций классов TSpisok и TDinMas для хранения данных.

Примерные варианты проектов.

1. Автоматическая составлялка расписаний. Интерфейс на основании Visual Studio. Входные данные: список аудиторий и время их доступности; список преподавателей и время их готовности вести занятия. Выходные данные: расписание занятий преподавателей по аудиториям.
2. Двухмерная аркада. Пользователь, управляя одним из объектов на поле, должен выполнить какое-то задание, чтобы пройти текущий уровень и перейти на следующий.
3. Логическая игра. Пошаговая игра. Описание алгоритмов игры компьютера на уровне пользователя.

Раздел 9. Итоговое занятие. (6 ч.)

Обсуждение итогов работы за год. Обсуждение задания на летний период. Итоговая диагностика.

Методическое обеспечение программы

Для реализации программы используются следующие формы и методы обучения.

Формы обучения: лекция, практикум, работа со специальной литературой, мини-конференция, обсуждение вариантов решения задачи.

Методы обучения:

1. Словесные (указания педагога, объяснение нового материала (лекции), индивидуальная консультация).
2. Практическая работа (задания, тесты, составление алгоритмов и схем, решение задач, наблюдение, проведение экспериментов, работа с литературными источниками).
3. Наблюдение (фото и видеосъёмка, проведение замеров).
4. Исследовательский (постановка, проведение и обработка результатов опытов и экспериментов, установление причинно-следственных связей).
5. Проблемного обучения (самостоятельный поиск учащимися ответа на поставленную проблему).
6. Иллюстративно-демонстративные (показ, пример, видеоиллюстрация).

Система оценки и фиксирования образовательных результатов

В процессе обучения осуществляется контроль за уровнем сформированности знаний, умений и навыков.

Система контроля за усвоением учащимися программы складывается из следующих элементов: опрос, зачеты, самостоятельные работы, соревнования (где можно определить уровень каждого игрока и команды), конкурсы, тесты. Результаты проверки уровня усвоения программы фиксируются педагогом в специально разработанных листах учебных достижений:

В течении учебного года по определению уровня усвоения программы обучающимися осуществляется три диагностических среза:

- **входная диагностика** посредством бесед, анкетирования, тестов, где выясняется начальный уровень знаний, умений и навыков обучающихся, а так же выявляются их творческие способности.
- **промежуточная диагностика** позволяет выявить достигнутый на данном этапе уровень ЗУН учащихся, в соответствии с пройденным материалом программы. Предлагаются контрольные тесты, выполнение практических заданий.
- **итоговая диагностика** проводится в конце учебного года (итоговый показ творческих проектов) и предполагает комплексную проверку образовательных результатов по всем ключевым

направлениям. Данный контроль позволяет проанализировать степень усвоения программы учащимися.

Результаты контроля фиксируются в диагностической карте.

Диагностика результативности образовательного процесса

Система оценки и фиксирования образовательных результатов

В процессе обучения осуществляется контроль за уровнем сформированности знаний, умений и навыков.

Система контроля за усвоением учащимися программы складывается из следующих элементов: опрос, зачеты, самостоятельные работы, соревнования (где можно определить уровень каждого игрока и команды), конкурсы, тесты. Результаты проверки уровня усвоения программы фиксируются педагогом в специально разработанных листах учебных достижений:

В течение учебного года по определению уровня усвоения программы учащимися осуществляется три диагностических среза:

- **входная диагностика** посредством бесед, анкетирования, тестов, где выясняется начальный уровень знаний, умений и навыков учащихся, а так же выявляются их творческие способности.
- **промежуточная диагностика** позволяет выявить достигнутый на данном этапе уровень ЗУН учащихся, в соответствии с пройденным материалом программы. Предлагаются контрольные тесты, выполнение практических заданий.
- **итоговая диагностика** проводится в конце учебного года (итоговый показ творческих проектов) и предполагает комплексную проверку образовательных результатов по всем ключевым направлениям. Данный контроль позволяет проанализировать степень усвоения программы учащимися.

Результаты контроля фиксируются в диагностической карте.

Виды контроля

Виды контроля	Содержание	Методы	Сроки контроля
Предварительный	Начальный уровень подготовки учащихся, имеющиеся знания, умения и навыки, связанные с предстоящей деятельностью.	Наблюдение, анкетирование, тестирование.	Сентябрь
Текущий	Освоение учебного материала по темам.	Проверочные задания по отдельным разделам программы.	Октябрь-апрель
Промежуточный	Освоение учебного материала за полугодие	Опрос, практическое задание	Декабрь-январь
Итоговый		Разработка итогового задания по предложенной теме.	Май

Промежуточная диагностика
по дополнительной общеобразовательной программе
«Основы ООП»

Педагог д/о _____
Группа № _____год обучения _____
Форма проведения _____

№ п/п	ФИ обучающегося	Количество баллов
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		
9.		
10.		

Низкий уровень –
учащийся со значительной помощью педагога ориентируется в содержании учебного материала и дает определение понятиям; освоил отдельные навыки и умения (1-2 балла).

Средний уровень –
почти полное усвоение учебного материала, принимает старательное участие в ответах на вопросы и в заданиях, иногда требуется помощь педагога. Учащийся старателен, внимательно слушает, но ответы нуждаются в уточнении; допускает неточности в работе (3-4 балла).

Высокий уровень –
учащийся самостоятельно ориентируется в содержании пройденного учебного материала, принимает активное участие в ответах на вопросы, полное усвоение содержания учебного материала; способен дать оценку собственной работе, умеет применять теоретические знания и практические умения и навыки в самостоятельной работе (5 баллов).

Средний балл _____

Оценка уровней освоения программы

Уровни / количество баллов	Параметры	Показатели
Высокий уровень/ 5 баллов	Теоретические знания.	Обучающийся освоил материал в полном объеме. Знает и понимает значение терминов, самостоятельно ориентируется в содержании материала по темам. учащийся заинтересован, проявляет устойчивое внимание к выполнению заданий.
	Практические умения и навыки.	Способен применять практические умения и навыки во время выполнения самостоятельных заданий. Правильно и по назначению применяет инструменты. Работу аккуратно доводит до конца. Может оценить результаты выполнения своего задания и дать оценку работы своего товарища.
	Конструкторские способности.	Учащийся способен узнать и выделить объект (алгоритм, модуль, элемент интерфейса). Учащийся способен собрать объект из готовых частей или построить с помощью инструментов. Учащийся способен выделять составные части объекта. Учащийся способен видоизменить или преобразовать объект по заданным параметрам. Учащийся способен из преобразованного или видоизмененного объекта, или его отдельных частей собрать новый.
Средний уровень/ 3-4 балла	Теоретические знания.	Учащийся освоил базовые знания, ориентируется в содержании материала по темам, иногда обращается за помощью к педагогу. Учащийся заинтересован, но не всегда проявляет устойчивое внимание к выполнению задания.
	Практические умения и навыки.	Владеет базовыми навыками и умениями, но не всегда может выполнить самостоятельное задание, затрудняется и просит помощи педагога. В работе допускает небрежность, делает ошибки, но может устранить их после наводящих вопросов или самостоятельно. Оценить результаты своей деятельности может с подсказкой педагога.
	Конструкторские способности.	Учащийся может узнать и выделить объект (алгоритм, модуль, элемент интерфейса). Учащийся не всегда способен самостоятельно разобрать, выделить составные части конструкции. Учащийся не способен видоизменить или преобразовать объект по заданным параметрам без подсказки педагога.
Низкий уровень / 1-2 балла	Теоретические знания.	Владеет минимальными знаниями, ориентируется в содержании материала по темам только с помощью педагога.

	Практические умения и навыки.	Владеет минимальными начальными навыками и умениями. Учащийся способен выполнять каждую операцию только с подсказкой педагога или товарищей. Не всегда правильно применяет необходимый инструмент или на использует вовсе. В работе допускает грубые ошибки, не может их найти их даже после указания. Не способен самостоятельно оценить результаты своей работы.
	Конструкторские способности.	Учащийся с подсказкой педагога может узнать и выделить объект (конструкцию, устройство). Учащийся с подсказкой педагога способен выделять составные части объекта. Разобрать, выделить составные части конструкции, видоизменить или преобразовать объект по заданным параметрам может только в совместной работе с педагогом.

Сводная таблица результатов обучения
по дополнительной общеобразовательной программе
«Основы ООП»

педагог д/о _____.

№ п/п	ФИ обучающегося	Оценка теоретических знаний	Оценка практических умений и навыков	Способности к программированию	Средний балл
1.					
2.					
3.					
4.					
5.					
6.					
7.					
8.					
9.					
10.					

Средний балл _____

Условия реализации программы

В качестве базового для изучения принципов процедурного и объектно-ориентированного программирования выбран язык C++. Он лежит в основе множества современных систем быстрой разработки программного обеспечения и является прототипом для ряда других современных языков ООП более высокого уровня (например, для C++), позаимствовавших у него свой синтаксис и некоторые возможности. Таким образом, изучение ООП на примере языка C++ позволит обучающимся в дальнейшем самостоятельно осваивать многие другие языки программирования.

Для разработки и отладки программ используется интегрированная среда разработки Visual Studio, которая обладает достаточной простотой использования и значительными возможностями. Используемый указанной средой компилятор соответствует всем стандартам языка C++. Библиотека, используемая Visual Studio, может быть существенно расширена за счёт добавления к ней собственных компонентов и компонентов сторонних разработчиков, свободно распространяемых по сети Internet. Расширения языка C++, вносимые средой разработки, во многом схожи с некоторыми стандартными возможностями других современных языков программирования, что существенно упростит в будущем знакомство с этими языками.

Занятия по данной программе проводятся группой и состоят из теоретической и практической части. Теоретическая часть проходит в виде лекций, практическая часть, как правило, содержит подробный разбор задачи; в зависимости от темы задачи могут быть предложены на самостоятельный разбор во время занятия.

На занятиях применяются технологии личностно-ориентированного, диалогового обучения. Для более интересного изучения введения в ООП перед разделом 6 «Основы объектно-ориентированного программирования» вводится раздел 5 «Графика»: с помощью графики и ООП строятся простые игры.

В ходе первого года обучения обучающиеся под руководством педагога разрабатывают итоговый проект. Тема выбирается и обсуждается с педагогом. После утверждения педагогом темы обучающийся самостоятельно разрабатывает свой проект, как правило, в виде элементарных игр, таких как «Змейка», «Стрелок», с использованием элементов объектно-ориентированного подхода. В ходе работы над проектом обучающийся обсуждает с педагогом возникающие вопросы.

В ходе второго года обучения каждый обучающийся самостоятельно разрабатывает проект по теме, обсуждённой и утверждённой педагогом. Темы проектов предлагаются в начале учебного года (раздел 1), темы для разработки выбираются во втором полугодии (раздел 8). Проекты, разрабатываемые обучающимися, могут быть выдвинуты на конкурс.

Материально-техническое обеспечение программы

Для реализации дополнительной образовательной программы «Основы ООП» необходимо иметь:

- На рабочих местах учащихся должны быть обеспечены уровни искусственной освещенности люминесцентными лампами при общем освещении помещений не ниже:

  в учебных помещениях для теоретических занятий - 300 - 500 лк;

  в компьютерных кабинетах - 300 - 500 лк;
- рабочие столы,
- доска демонстрационная,
- шкафы и стеллажи для хранения техники и конструкторов.

Оборудование и ПО

- персональные компьютеры;
- ПО Visual Studio;
- проектор или интерактивная доска;
- устройство для 3d-печати.

Медицинская аптечка для оказания доврачебной помощи.

Список литературы

Литература для обучающихся

1. Архангельский А. Я. C++Builder 6. Справочное пособие. Кн. 1. Язык C++. – М.: Бином-Пресс, 2002. – 544 с., ил.
2. Архангельский А. Я. C++Builder 6. Справочное пособие. Кн. 2. Классы и компоненты. – М.: Бином-Пресс, 2002. – 528 с., ил.
3. Вандервурд Д., Джосаттис Н. М. Шаблоны C++: Справочник разработчика / Пер. с англ. – М.: Изд. дом «Вильямс», 2003. – 544 с., ил.
4. Златопольский, Д. М. Сборник задач по программированию / Д. М. Златопольский.– 2-е изд.– Санкт-Петербург : БХВ-Петербург, 2007.– 240 с.
5. Кнут Д. Э. Искусство программирования. Т. 1. Основные алгоритмы. 3-е изд. / Пер. с англ. – М.: Изд. дом «Вильямс», 2001. – 720 с., ил.
6. Кнут Д. Э. Искусство программирования. Т. 3. Сортировка и поиск. 2-е изд. / Пер. с англ. – М.: Изд. дом «Вильямс», 2001. – 832 с., ил.
7. Коплиен Дж. Программирование на C++. Классика CS. – СПб.: Питер, 2005. – 479 с., ил.
8. Стивен Прата Язык программирования C++. Лекции и упражнения, 5-е издание / М.: Вильямс, 2007. – 1248 с.
9. Харви Дейтел, Пол Дейтел Как программировать на C++: Третье издание. Пер с англ. - М.: ЗАО "Издательство БИНОМ", 2003. – 1011 с.

Литература для педагога

1. Александров Э. Э. Программирование на языке С в MicrosoftVisualStudio2010 : учеб. пособие / Э. Э. Александров, В. В. Афонин. – М. : Изд-во Жарков Пресс, 2010. – 424 с.
2. Архангельский А. Я. C++Builder 6. Справочное пособие. Кн. 1. Язык C++. – М.: Бином-Пресс, 2002. – 544 с., ил.
3. Архангельский А. Я. C++Builder 6. Справочное пособие. Кн. 2. Классы и компоненты. – М.: Бином-Пресс, 2002. – 528 с., ил.
4. Вандервурд Д., Джосаттис Н. М. Шаблоны C++: Справочник разработчика / Пер. с англ. – М.: Изд. дом «Вильямс», 2003. – 544 с., ил.
5. Горячев, А.В. Информатика в играх и задачах. / А.В. Горячев, К.И Горина, Н.И. Суворова. – М.: Баласс, 2009. – 112 с.
6. Кнут Д. Э. Искусство программирования. Т. 1. Основные алгоритмы, 3-е изд. / Пер. с англ. – М.: Изд. дом «Вильямс», 2001. – 720 с., ил.
7. Кнут Д. Э. Искусство программирования. Т. 3. Сортировка и поиск. 2-е изд. / Пер. с англ. – М.: Изд. дом «Вильямс», 2001. – 832 с., ил.
8. Коплиен Дж. Программирование на C++. Классика CS. – СПб.: Питер, 2005. – 479 с., ил.
9. Матвеев А.С. Дополнительная общеразвивающая программа «Современное программирование» / А.С. Матвеев, Р.А. Кабетов, Ф.А. Вомпе, Ю.А. Пюрбеев // Образовательные программы дополнительного образования детей (приложение к журналу «Дополнительное образование и воспитание»).- 2014.- №5.-С.31-60.- Возраст обучающихся – 12-17 лет, срок реализации программы – 2 года, 144 часа в год.
10. Подбельский В. В. Язык C++: Учеб. пособие. 4-е изд. – М.: Финансы и статистика, 1999. – 560 с., ил.
11. Пол И. Объектно-ориентированное программирование с использованием C++ / Пер. с англ. – К.: НИПФ «ДиаСофт Лтд», 1995. – 480 с.

12. C++. Стандартная библиотека. Для профессионалов / Н. Джосьютис. – СПб.: Питер, 2004. – 730 с., ил.
13. Страуструп Б. Язык программирования C++. Специальное издание / Пер. с англ. – М.: -Пресс», 2006. – 1104 с., ил.
14. Уоррен Г. С. Алгоритмические трюки для программистов. Испр. изд. / Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 288 с., ил.
15. Холингворт Д., Сворт Б., Кэшмэн М., Густавсон П. Borland C++Builder 6. Руководство разработчика / Пер. с англ. – М.: Изд. дом «Вильямс», 2003. – 276 с., ил.

Приложения

Первичный тест (1 год обучения)

«Основы алгоритмизации»

1. **Алгоритм - это**
   а**)** правила выполнения определенных действий;
   б) предписание исполнителю совершить последовательность действий, направленных на достижение поставленных целей;
   в) набор команд для компьютера.
2. **Какой из документов является алгоритмом?**
   а) Правила техники безопасности.
   б) Инструкция по получению денег в банкомате.
   в) Расписание уроков.
3. **Какой из объектов может являться исполнителем?**
   а) Луна. б) Карта. в) Принтер. г) Книга
4. **Дискретность- свойство алгоритма означающее…**
   а) однозначность правил выполнения алгоритма
   б) правильность результатов выполнения алгоритма
   в) деление алгоритма на отдельные шаги
5. **Свойством алгоритма является*:***
   а) конечность;
   б) цикличность;
   в) возможность изменения последовательности команд;
   г) возможность выполнения алгоритма в обратном порядке.
6. **Алгоритм называется линейным, если*:***
   а) он составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
   б) ход его выполнения зависит от истинности тех или иных условий;
   в) его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий.
7. **Алгоритм структуры «ветвление» предусматривает**
   а) выбор условий, б) выбор алгоритмов, в) выбор команд (действий)
8. **Алгоритм называется циклическим, если:**
   а) он составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
   б) ход его выполнения зависит от истинности тех или иных условий;
   в) его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий.
9. **Алгоритм называется вспомогательным, если**
   а) он предполагает выбор действий
   б) повторяет действия до выполнения какого – либо условия;
   в) решает часть задачи и вызывается из основной программы.
10. **Цикл со счётчиком**
    а) зависит от некоторого условия; б) зависит от известного числа повторений.
11. **Какой тип алгоритмической структуры необходимо применить, если последовательность команд выполняется или не выполняется в зависимости от условия**
    а) цикл б) ветвление в) линейный.
12. **Ромб — графический объект, используемый в блок-схеме для записи:**
    а) ввода, вывода данных; б) вычислительных действий;
    в) конца выполнения задачи; г) условия выполнения действий.

13. **Переменная для компьютера – это**
    а) буква алфавита б) различные числа в) область памяти

Ключ к тесту

1	**б**
2	**б**
3	**в**
4	**в**
5	**а**
6	**в**
7	**в**
8	**а**
9	**в**
10	**б**
11	**б**
12	**Г**
13	**в**

Первичный тест (2 год обучения)

«Основы с++»

1. Какой служебный знак ставится после оператора `case` ?

- `-`
- `:`
- `;`
- `.`

2. До каких пор будут выполняться операторы в теле цикла `while (x < 100)`?

- Пока `x` меньше или равен стам
- Пока `x` строго меньше ста
- Пока `x` больше ста
- Пока `x` равен стам

3. Укажите объектно-ориентированный язык программирования

- Все варианты ответов
- C++
- Java
- Eiffel

4. Какой из ниже перечисленных операторов, не является циклом в C++?

- `while`
- `repeat until`
- `for`
- `do while`

5. Программа, переводящая входную программу на исходном языке в эквивалентную ей выходную программу на результирующем языке, называется:

- компилятор
- интерпретатор
- сканер
- транслятор

6. Какой оператор не допускает перехода от одного константного выражения к другому?

- `break;`
- точка с запятой
- `Stop;`
- `end;`

7. Цикл с постусловием?

- `for`
- `do while`
- `while`

8. В приведённом коде измените или добавьте один символ чтобы код напечатал 20 звёздочек (`*`).

```
int i, N = 20;
for(i = 0; i < N; i--)
    printf("*");
```

```
int i, N = 20;
for(i = 19; i < N; i--)
    printf("*");
```

```
int i, N = 40;
for(i = 0; i < N; i--)
    printf("*");
```

```
int i, N = 20;
for(i = 0; i < N; N--)
    printf("*");
```

```
int i, N = 20;
for(i = 20; i < N; i--)
    printf("*");
```

9. Какую функцию должны содержать все программы на C++?

- `start()`
- `system()`
- `main()`
- `program()`

10. Какому зарезервированному слову программа передаёт управление в случае, если значение переменной или выражения оператора `switch` не совпадает ни с одним константным выражением?

- `all`
- `default`
- `other`
- `contingency`

11. Какой из следующих операторов - оператор сравнения двух переменных?

- `equal`
- `:=`
- `==`
- `=`

12. Тело любого цикла выполняется до тех пор, пока его условие ...

- истинно
- ложно

- у цикла нет условия

13. Что будет напечатано?

```
1
2 int main()
3  {
4     for (int i = 0; i < 4; ++i)
5     {
6        switch (i)
7        {
8           case 0  : std::cout << "0";
9           case 1  : std::cout << "1"; continue;
10          case 2  : std::cout << "2"; break;
11          default : std::cout << "D"; break;
12       }
13       std::cout << ".";
14    }
15    return 0;
   }
```

- 01.2.D.
- Ошибка компиляции в строке 10
- 011.2.D
- 0112.D.
- 0.1.2.

14. Чему будет равна переменная `a`, после выполнения этого кода `int a; for(a = 0; a < 10; a++) {}`?

- 10
- 9
- 1

15. Какая из следующих записей - правильный комментарий в C++?

- */ Комментарии */
- {комментарий}
- ** Комментарий **
- /* комментарий */

16. Укажите правильную форму записи цикла `do while`

- ```
  1// форма записи оператора цикла do while:
  2do // начало цикла do while
  3{
  4/*блок операторов*/;
  5}
  6while (/*условие выполнения цикла*/); // конец цикла do while
  ```
- ```
  1// форма записи оператора цикла do while:
  2do // начало цикла do while
  3{
  4/*блок операторов*/;
  5}
  6while (/*условие выполнения цикла*/) // конец цикла do while
  ```
-

```
// форма записи оператора цикла do while:
do // начало цикла do while
{
/*блок операторов*/;
}
while {/*условие выполнения цикла*/} // конец цикла do while
```

17. Общий формат оператора множественного выбора - switch

- ```
  switch (switch_expression)
  {
     case constant1, case constant2: statement1; [break;]
     case constantN: statementN; [break;]
     [default: statement N+1;]
  }
  ```
- ```
  switch (switch_expression)
  {
      case constant1: statement1; [break;]
      case constant2: statement2; [break;]
      case constantN: statementN; [break;]
      [else: statement N+1;]
  }
  ```
- ```
  switch (switch_expression)
  {
      case constant1: statement1; [break;]
      case constant2: statement2; [break;]
      case constantN: statementN; [break;]
      [default: statement N+1;]
  }
  ```

18. Каков результат работы следующего франмента кода?

```
int x = 0;

    switch(x)
    {

      case 1: cout << "Один";

      case 0: cout << "Нуль";

      case 2: cout << "Привет мир";

    }
```

- Привет мир
- Нуль
- Один
- НульПривет мир

19. Какие среды программирования (IDE) предназначены для разработки программных средств?
```

- MVS, NetBeans, QT Creator, RAD Studio, Dev-C++
- MVS, Code::Blocks, QT Creator, AutoCAD, Eclipse
- MVS, Code::Blocks, QT Creator, RAD Studio, MathCAD

20. Простые типы данных в C++.
    - целые – `bool`, вещественные – `float` или `double`, символьные – `string`
    - целые – `int`, вещественные – `float` или `double`, символьные – `string`
    - целые – `int`, вещественные – `float` или `real`, символьные – `char`
    - целые – `int`, вещественные – `float` или `double`, символьные – `char`
21. Укажите правильное определение функции `main` в соответствии со спецификацией стандарта ANSI

```
1void main()
1int main(void)
1void main(void)
1int main()
```

22. Чтобы подключить заголовочный файл в программу на C++, например `iostream` необходимо написать:
    - `#include <>` с `iostream` внутри скобок
    - `#include <>;` с `iostream.h` внутри скобок
    - `include (iostreamh)`
    - `include #iostream,h;`
23. Какой из перечисленных типов данных не является типом данных в C++?
    - `real`
    - `double`
    - `int`
    - `float`
24. Какое значение, по умолчанию, возвращает программа операционной системе в случае успешного завершения?
    - Программа не возвращает значение.
    - -1
    - 1
    - 0
25. Цикл с предусловием?
    - `do while`
    - `for`
    - `while`
26. Выберите правильный вариант объявления константной переменной в C++, где `type` - тип данных в C++ `variable` - имя переменной `value` - константное значение
    - `const type variable = value;`
    - `const type variable := value;`
    - `const variable = value;`
27. Какими знаками заканчивается большинство строк кода в Си++?
    - `:` (двоеточие)
    - `;` (точка с запятой)
    - `.` (точка)
    - `,` (запятая)
28. Название C++ предложил
    - Рик Масситти
    - Дональд Кнут

- Бьерн Страуструп
- Кэн Томпсон

29. Структура объявления переменных в С++
    - `[=];< идент. 2>,…;`
    - `[==]; < идент. 2>,…;`
    - `[:=], < идент. 2>,…;`
    - `[=], < идент. 2>,…;`
30. Какие служебные символы используются для обозначения начала и конца блока кода?
    - `( )`
    - `begin end`
    - `{ }`
    - `< >`
31. Язык программирования С++ разработал
    - Дональд Кнут
    - Бьерн Страуструп
    - Никлаус Вирт
    - Кен Томпсон

Ключ

1	2	**18**	2
2	2	**19**	1
3	3	**20**	4
4	4	**21**	3
5	4	**22**	2
6	2	**23**	3
7	3	**24**	1
8	1	**25**	3
9	3	**26**	1
10	2	**27**	2
11	3	**28**	1
12	1	**29**	4
13	2	**30**	3
14	1	**31**	4
15	2		
16	3		
17	1		

Срез знаний учащихся в середине учебного года (1 год обучения)

Написать программу, в которой генерируется строка символов заданного размера (более трех) и для которой определяется подстрока из трех символов, вводимая пользователем. В случае, когда подстрока не обнаружена, предусмотреть генерирование случайной строки поиска 1 000 раз, и программа должна искать подстроку до первого совпадения.

Программный код решения примера

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

int main (void) {
        int i, j, k, n, in,
                N = 1000,
                numA, numZ,
            ch_box[3];
        char str[16],
                str2[4];

        srand((unsigned)time(NULL));
        numA = (int)'a'; // числовой код латинской буквы a
        numZ = (int)'z'; // числовой код латинской буквы z

        printf("\n Enter a string of 3 letters: ");
        in = scanf_s("%c", str2, sizeof(str2));
        if ( in == 0 )  {
                printf("\n Error input. Press any key: ");
                getch();
                exit(1);
        }
        printf("\n\t substring is \"%c\"\n", str2);


    for (n = 0; n < N; n++) {
                for (i = 0; i < 15; i++)
                        str[i] = numA + rand() % (numZ - numA) + 1;
                str[i] = '\0';
                k = 0;
                for (i = 0; i < 13; i++)
if (str2[0] == str[i] && str2[1] == str[i+1] && \
   str2[2] == str[i+2])
                        {
ch_box[0] = i; ch_box[1] = i+1; ch_box[2] = i+2;
    k++;
```

```
        break; }
                        if (k > 0)
                                break;
        }

        if (k == 0)
                printf("\n\t \"%c\" not found", str2);
        else
 printf("\n Substring \"%c\" found at positions %d, %d, %d", \
 str2, ch_box[0]+1, ch_box[1]+1, ch_box[2]+1);

        puts("\n");
        for (k = 0; k < 15; k++)
                printf(" %3d)", k+1);
        puts("");
        for (j = 0; j < 15; j++)
                printf(" %3c ", str[j]);

        printf("\n\n ... Press any key: ");
        getch();
        return 0;
}
```

Срез знаний учащихся в середине учебного года (2 год обучения)

Написать программу по решению задачи «Ханойская башня». Имеется пирамида из *n* колец, лежащих на основании *A* (на стержне, на столбе) одно на другом в порядке убывания размеров. Кольца должны быть перемещены на основание *B* в том же порядке, в котором они были на основании *A*, при использовании промежуточного основания *C*. Единственными разрешенными перемещениями являются такие, при которых кольцо, взятое с вершины одной из пирамид, помещается на большее кольцо, либо на пустое основание. Осуществить выдачу на печать (на консоль, в текстовый файл) последовательности перемещений колец.

Программный код решения примера

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

// Пртотип рекурсивной функции
void hanoi(int n,char a, char b, char c);

int main(void) {
    char a = 'A'; // исходное основание
    char b = 'B'; // конечное основание
    char c = 'C'; // промежуточное основание


        double n; // число дисков
        char str[81] = "15"; // инициализация строки

puts("\n Hanoi Tower; A - starting; B - end; C - intermediate");

        if (0.01) {      // 0.01 - как истинное значение
                printf("\n Enter the number of disks: ");
                scanf_s(" %c",str, 80);
                n = atof(str);

                if ( n <= 0 || ceil(n) != n) {
                        printf("\n Error input. Press any key: ");
                        getch();
                        continue;
                }

                else if (n > 15.0) {
                        printf("\n Very large number. Press any key: ");
                        getch();
                        continue;
                }

      else
                        break;
```

```
        }

    puts("\n Elementary transferences:");
        hanoi((int)n, a, b, c);


    printf("\n\n Press any key: ");
    getch();
    return 0;
}

// Определение рекурсивной функции
void hanoi(int num, char a, char b, char c)
{
  if(num > 0) {

    hanoi(num-1,a,c,b);

    printf("\t%c --> %c\n",a,b);

    hanoi(num-1,c,b,a);

  }

}
```

Итоговое задание.

1 год обучения.

Вычисление последовательности Фибоначчи с использованием больших чисел

В языке программирования **C** существует большое количество целочисленных типов данных, области определения которых охватывают числа различных диапазонов. При решении задачи программист должен выбрать правильные типы данных для числовых переменных, исходя из природы соответствующих данных, а также требований к занимаемой программой и ее данными памяти и желаемому быстродействию. Тем не менее диапазоны встроенных целочисленных типов данных языка **C** ограничены, что не позволяет выполнять вычисления над достаточно большими числами (например, с разрядностью более 20 десятичных цифр). Это оказывается серьезным препятствием при решении задач, требующих выполнения операций над большими числами, например вычисления элементов быстрорастущих последовательностей. В частности, использование типа **int** языка **C** для вычисления элементов последовательности Фибоначчи позволяет получить только 46 первых чисел, после чего наступает целочисленное переполнение.

Проблема ограниченности диапазонов целочисленных типов данных решается при помощи так называемой длинной арифметики, под которой в вычислительной технике понимают операции над числами, разрядность которых превышает длину машинного слова данной вычислительной машины. На практике длинная арифметика применяется в следующих случаях:

- на компьютерах с процессорами низкой разрядности и микроконтроллерах. Например, на компьютерах и микроконтроллерах с 8-битными процессорами без использования длинной арифметики невозможно выполнить никакие сколько-нибудь полезные вычисления;
- при решении задач криптографии;
- при создании математического и финансового программного обеспечения, требования к точности вычислений в котором очень высоки и критичны, а ошибки округления и переполнения недопустимы;
- для «спортивных» вычислений трансцендентных чисел (π, e и т. д.) с высокой точностью;
- для решения олимпиадных задач по программированию.

Рассмотрим, каким образом можно хранить длинные целые числа в памяти компьютера и как выполнять над ними действия. Обычно длинное число представляют в виде массива, элементы которого содержат цифры длинного числа, и отдельно дополнительно сохраняют длину числа, т. е. количество значимых цифр. В этом случае удобнее перейти от десятичной системы счисления к системе счисления с большим основанием, так как появится возможность лучше использовать пространство оперативной памяти, занятой массивом. Количество значимых цифр занесем в первый элемент массива (с индексом 0). Цифры числа будем хранить в обратном порядке, т. е. младшая цифра будет храниться в элементе массива с меньшим индексом.

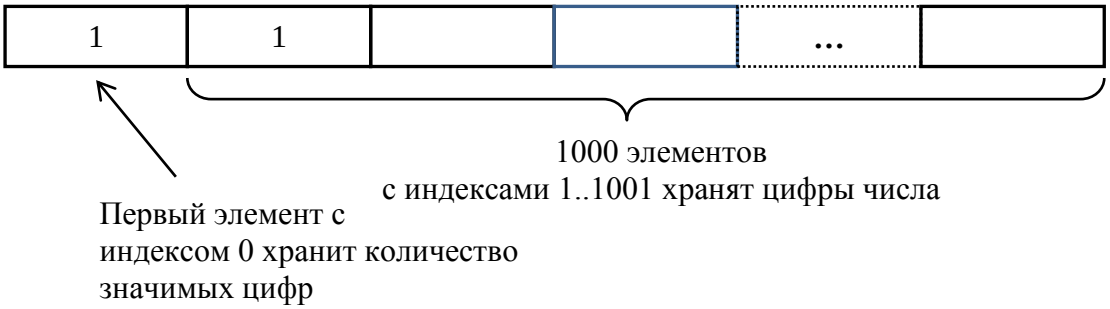
Сделаем следующие объявления:

```
/// максимальная длина числа - 1000 знаков
#define NUMMAX          1000
/// основание системы счисления длинных чисел - 10^9
#define NUMBASE         1000000000
```

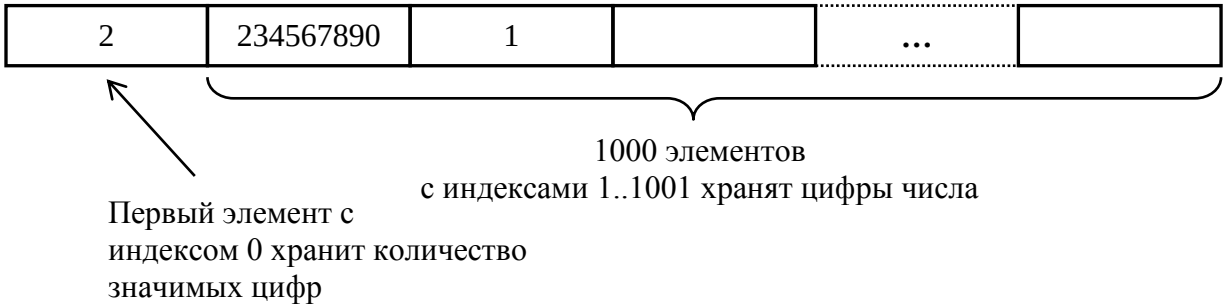
```
/// получение длины числа
#define NUMLEN(n)        ((n)[0])
/// определение типа длинных чисел
typedef int number_t[NUMMAX + 1];
```

Итак, мы определили тип данных длинных чисел – **number_t**. С его помощью можно хранить и обрабатывать числа длиной до 9000 десятичных знаков. Основание системы счисления выбрано равным 10^9, это позволяет содержать в одном элементе массива до 9 десятичных знаков. Рассмотрим пример хранения двух чисел: 1 и 1234567890.

Число 1 меньше выбранного основания системы счисления, поэтому оно будет представлено одной значимой цифрой и для его хранения будет достаточно 1 элемента массива. Схема описания может быть следующей:



Число 1234567890 больше выбранного основания системы счисления, поэтому будет представлено двумя значимыми цифрами и для его хранения потребуются 2 элемента массива. Схема описания может быть следующей:



Создадим три вспомогательные функции, задающие значение длинного числа. Функция **numzero()** устанавливает длинное число равным нулю. Ее программный код выглядит так:

```
/// Сброс длинного числа в ноль
void numzero (number_t lhs)
{
        lhs[0] = 1;
        lhs[1] = 0;
}
```

Функция **numassgns()** присваивает длинному числу короткое значение из диапазона $0..2^{32}–1$. Программный код функции имеет вид

```
/// Присваивание короткого числа длинному числу
void numassgns (number_t lhs, unsigned rhs)
{
        lhs[0] = 0;
```

```
        if (rhs)
        {
                lhs[++lhs[0]] = rhs % NUMBASE;
                rhs /= NUMBASE;
        }
}
```

Функция **numassgn()** присваивает одному длинному числу значение другого длинного числа. Ее программный код таков:

```
/// Присваивание длинных чисел
void numassgn (number_t lhs, const number_t rhs)
{
        int i;

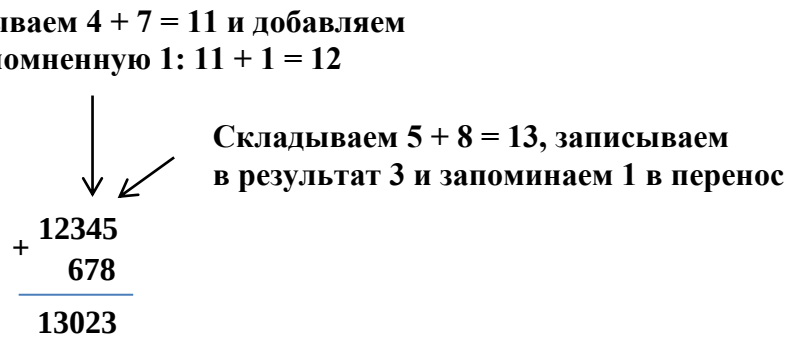
        // Переписываем длину результирующего числа
        lhs[0] = rhs[0];
        // Копируем цифры
        for (i = 1; i <= NUMLEN (rhs); ++i)
                lhs[i] = rhs[i];
}
```

Для вывода длинного числа на экран создадим функцию **numprint()**, которая сначала проверяет длину числа, и если она равна 0, то выводит 0, в противном случае печатает цифры числа на экране проходя по массиву в обратном направлении – от индексов с большими значениями до индексов с меньшими, так как цифры числа хранятся в обратном порядке. Программный код функции:

```
/// Печать длинного числа
void numprint (const number_t lhs) {
        int i;

        printf ("%d", NUMLEN (lhs) ? lhs[NUMLEN (lhs)] : 0);
        for (i = NUMLEN(lhs) - 1; i > 0; --i)
                printf ("%09d", lhs[i]); }
```

Операция сложения длинных чисел реализует обычное сложение чисел столбиком. Вспомним, как она выполняется. Пусть надо сложить два числа 12345 и 678. Записываем их в столбик таким образом, чтобы младшие разряды числа оказались друг под другом. После этого по таблице сложения складываем независимо разряды друг с другом. Если результат превосходит 9, то запоминаем 1, переносим ее в старший разряд, а в текущем записываем младший разряд от результата сложения. И так продолжается до тех пор, пока все разряды не будут учтены. Обратите внимание, что если длина чисел разная, то в старших разрядах более длинного числа сложение производится только с «запомненной» 1 переноса. Схема вычислений может быть следующей:



Сложение чисел выполняется функцией **numadd()**, которая принимает два числа – слагаемые и записывает результат в параметр с именем res. Данная функция выбирает из двух слагаемых более короткое и сначала складывает разряды двух чисел, а затем, когда все разряды более короткого числа будут учтены, добавляет оставшиеся разряды более длинного числа с учетом переноса, признак которого хранится в переменной **c**. Так как цифры числа хранятся в обратном порядке, то сложение осуществляется по возрастанию индексов массива. Программный код функции будет таким:

```
/// Сложение длинных чисел
void numadd (number_t res, const number_t lhs, const number_t rhs)
{
	int i = 0;
	// Флаг переноса
	int c = 0;
	// Число с минимальной длинной
	const int *sn = NUMLEN (lhs) < NUMLEN (rhs) ? lhs : rhs;
	// Число с максимальной длиной
	const int *ln = sn == lhs ? rhs : lhs;

	// Складываем два числа
	if (i < NUMLEN (sn)) {
		++i;
		res[i] = c + sn[i] + ln[i];
		c = res[i] > NUMBASE ? 1 : 0;
		if (c) res[i] -= NUMBASE;
	}
	// Добавляем остаток от более длинного числа и перенос
	if (i < NUMLEN (ln))
	{
		++i;
		res[i] = c + ln[i];
		c = res[i] > NUMBASE ? 1 : 0;
		if (c) res[i] -= NUMBASE;
	}
	// Учитываем последний перенос
	if (c) res[++i] = c;
	// Сохраняем длину числа
	res[0] = i;}
```

Рассмотрим пример использования арифметики длинных чисел. Функция **main()** создает три переменные для хранения длинных чисел, инициализирует их значениями и выполняет операцию сложения, после чего печатает результат на экране. Программный код главной функции выглядет следуюшим образом:

```
int main (int argc, char** argv[])
{
        int i;

        number_t a, b, c;
        numassgns (a, 1234567890);
        numassgns (b, 1);
        numadd (c, a, b);
        numprint (c);

    printf("\n\n ... Press any key: ");
   getch();

        return 0;
}
```

Задание

1. Создайте функцию **numtoa()**, выполняющую преобразование длинного числа в строку. Функция должна иметь следующий прототип:

```
/// Перевод длинного числа в строку
void numtoa (const number_t num, char**str);
```

2. Создайте функцию **atonum()**, выполняющую преобразование строки в длинное число. Функция должна иметь такой прототип:

```
/// Перевод строки в длинное число
void atonum (const char**str, number_t num);
```

3. Разработайте программу, выполняющую вычисление 500 первых чисел последовательности Фибоначчи. Элементы последовательности Фибоначчи вычисляются по следующему рекуррентному соотношению:

$$a_i = a_{i-1} + a_{i-2},\ a_1 = 1,\ a_2 = 1.$$

2 год обучения.

Покупки в супермаркете

Сканер в кассе супермаркета выдает последовательность штрихкодов для товаров в корзине покупок, например

1234, 4719, 3814, 1112, 1111, 1111, 1234,

которая должна быть преобразована в чек следующего вида:

C supermarket

Dry Sherry, 1lt ..... (x2) $108.20
Fish Fingers ..... (x1) $12.11
Orange Jelly ..... (x1) $5.61
Giant Hula Hoops ..... (x1) $13.31
Hula Hoops ..... (x2) $4.22

TOTAL $143.45

Решение начнем с анализа данных. Для представления номенклатурной единицы супермаркета создадим структуру, хранящую данные о штрихкоде товара, его наименовании и цене. Цену зададим в центах. Определение структуры:

```
/// Информация о товаре магазина
struct Item {
        /// Штрихкод товара
        int m_barCode;
        /// Наименование товара, не более 85 символов
        char m_name[86];
        /// Цена 1 единицы товара в центах
        int m_price;
};
```

База данных (БД) товаров в простейшем случае – массив, который хранит структуры товара и количество номенклатурных позиций супермаркета. Опишем БД товаров в виде следующей структуры:

```
/// База данных товаров
struct ItemDatabase {
        /// количество товаров в БД
        int m_count;
        /// массив товаров
        struct Item *m_items;
};
```

Для поиска товара по его штрихкоду в БД создадим функцию **FindItem()**,способную принимать два аргумента: указатель на БД товаров и штрихкод товара, информацию по которому необходимо извлечь из БД:

```
/// Поиск товара в БД по штрихкоду
struct Item *
```

```
FindItem (const struct ItemDatabase *database, int barCode){
	int i;


	assert (database != NULL);
	assert (database->m_count > 0);

	for (i = 0; i < database->m_count; ++i)
		if (database->m_items[i].m_barCode == barCode)
			return database->m_items + i;
	return database->m_items;
}
```

Функция линейным поиском проходит по БД товаров и, если ей удается найти товар с соответствующим штрихкодом, возвращает указатель на структуру с информацией о нем. Если товара с таким штрихкодом нет, функция возвращает указатель на первый товар в БД.

Для представления товара в чеке введем дополнительную структуру, которая будет хранить пару «товар – количество товара в чеке»:

```
/// Строка счета
struct BillItem {
	/// Товар
	const struct Item *m_item;
	/// Количество в счете
	int m_quantity;
};
// Чек – это массив структур BillItem.
/// Счет
struct Bill {
	/// Количество товаров в счете
	int m_count;
	/// Массив товаров в счете
	struct BillItem *m_items;
};
```

Создадим две вспомогательные функции, которые будут осуществлять управление памятью для структуры чека. Функция **AllocBill()** выделяет память под структуру чека и резервирует память под массив товаров, функция **FreeBill()** освобождает всю память, связанную с чеком:

```
/// Выделяет память под счет
struct Bill * AllocBill (int itemsCount) {
	struct Bill *result;
	result = malloc (sizeof (struct Bill));
	if (!result)
		return NULL; // выделение памяти было неуспешным
	result->m_count = 0;
	result->m_items = malloc (itemsCount * sizeof (struct BillItem));
	if (!result->m_items && itemsCount > 0) {
		free (result);
		return NULL;
	}
```

```
	return result;
}

/// Освобождает память, связанную со структурой счете
void FreeBill (struct Bill *bill) {
	assert (bill != NULL);

	if (bill->m_items)
		free (bill->m_items);
	free (bill);
}
```

Для печати чека на экране создадим функцию **PrintBill()**. На входе она принимает указатель на печатаемый чек и выводит его на экран с соответствующим форматированием:

```
/// Печатает счет на экран
void PrintBill (const struct Bill *bill) {
	int i;
	int totalBill = 0;
	assert (bill != NULL);
// Печатаем заголовок
	printf ("C supermarket\n\n");
	// Печатаем список товаров
	for (i = 0; i < bill->m_count; ++i) {
int totalLine = bill->m_items[i].m_item->m_price*bill->m_items[i].m_quantity;

		totalBill += totalLine;
		printf ("%c (x%d) $%d.%d\n",
				bill->m_items[i].m_item->m_name,
				bill->m_items[i].m_quantity,
				totalLine / 100,
				totalLine % 100
			);
	}
	// Печатаем строку итого
printf ("\nTOTAL $%d.%d\n", totalBill / 100, totalBill % 100);
}
```

Центральной является функция **ProduceBill()**, которая по массиву штрихкодов создает структуру счета. Она сначала выделяет память под результат при помощи **AllocBill()**, затем проходит по списку переданных штрихкодов, ищет информацию о соответствующем товаре в БД и включает ее в счет. Если в счете уже имеется такой товар, то функция просто увеличивает его количество. Если товара нет, функция включает в счет новую строку:

```
/// Создает счет по списку товаров
struct Bill *
ProduceBill (const struct ItemDatabase *database, const int *barCodes, int count)
{
	int i;
	struct Bill *result;
```

```
        assert (database != NULL);

        assert (barCodes != NULL);
        result = AllocBill (count);
        if (!result)
                return NULL;
// Формируем счет
        for (i = 0; i < count; ++i) {
// Выполняем поиск элемента в счете, может он был добавлен ранее
                int index = IndexOfBillItem (result, barCodes[i]);
                if (index == -1) {
                        // Не нашли, добаляем новый элемент
result->m_items[result->m_count].m_item = FindItem (database, barCodes[i]);
                result->m_items[result->m_count].m_quantity = 1;
                        ++result->m_count;
                }
                else {
                        // Нашли, тогда увеличиваем количество
                        ++result->m_items[index].m_quantity;
                }
        }
        return result;
}
```

Для проверки того, есть ли товар с некоторым штрихкодом в счете, функция **ProduceBill()** использует специальную вспомогательную функцию **IndexOfBillItem()**, которая возвращает либо индекс товара в счете, либо число –1, если товара в счете нет. Программный код вспомогательной функции **IndexOfBillItem()**выглядет так:

```
/// Поиск товара в счете
int IndexOfBillItem (const struct Bill *bill, int barCode){
        int i;
        assert (bill != NULL);
        for (i = 0; i < bill->m_count; ++i)
                if (bill->m_items[i].m_item->m_barCode == barCode)
                        return i;
        return -1;
}
```

Базу данных товаров определим в виде константы с помощью спецификатора **static**:

```
/// База данных товаров в супермаркете
static struct Item g_databaseItems[] = {
        { 0, "Unknown item", 0 },
        { 4719, "Fish Fingers", 1211 },
        { 5643, "Nappies", 1010 },
        { 3814, "Orange Jelly", 561 },
        { 1111, "Hula Hoops", 211 },
        { 1112, "Giant Hula Hoops", 1331 },
        { 1234, "Dry Sherry, 1lt", 5401 }
};
```

```
const struct ItemDatabase g_database = {
        sizeof (g_databaseItems) / sizeof (g_databaseItems[0]),
        g_databaseItems
};
```

Функция **main()**создает и печатает счет:

```
int main (int argc, char** argv[])
{       int codes[] = { 1234, 4719, 3814, 1112, 1111, 1111, 1234 };
        struct Bill *bill = ProduceBill (&g_database, codes, sizeof (codes) / sizeof (codes[0]));
        if (!bill) {
                printf ("Error: out of memory if creating a bill");
                return -1;
        }

        PrintBill (bill);
        FreeBill (bill);
        return 0;
}
```

Задание

1. Произведите сборку программы и осуществите ее компиляцию.
2. *Форматирование счета*. Измените функцию **PrintBill()**, чтобы она печатала счет в соответствии с форматированием, приведенным в условии. Ширина строки счета – 40 символов.
3. *Отбрасывание неизвестных товаров*. Измените функцию **ProduceBill()** таким образом, чтобы товары, не найденные в БД, не включались в счет.
4. *Вычисление скидки*. За каждые две купленные бутылки шерри (код 1234) супермаркет дает скидку $5.00 с суммы счета. Добавьте вычисление скидки по счету. Результирующий счет должен выглядеть следующим образом:

C supermarket

Dry Sherry, 1lt ..... (x2) $108.20
Fish Fingers ..... (x1) $12.11
Orange Jelly ..... (x1) $5.61
Giant Hula Hoops ..... (x1) $13.31
Hula Hoops ..... (x2) $4.22

Discount $5.00

TOTAL $138.45

5. *Оптимизация поиска товара в БД*. Функция **FindItem()** имеет сложность O(N). Измените структуру БД или тело функции **FindItem()** таким образом, чтобы уменьшить алгоритмическую сложность поиска.
6. *Загрузка БД из файла*. Разработайте функции, выполняющие загрузку списка товаров супермаркета из CSV-файла. CSV-файлом (или файлом значений, разделенных запятыми,

Comma-separated values) называется текстовый файл, в котором содержатся записи, состоящие из нескольких полей. При этом каждая новая строка соответствует одной записи. Поля одной записи разделяются запятыми. Если значение поля не содержит запятых, то оно записывается непосредственно. Если в значении есть запятые, то оно заключается в двойные кавычки ("). Если в такой последовательности содержится двойная кавычка, она удваивается. Имеется разновидность формата, когда все строковые значения заключаются в кавычки. Пример БД товаров в CSV-файле:

4719,"Fish Fingers",1211
5643,"Nappies",1010
3814,"Orange Jelly",561
1111,"Hula Hoops",211
1112,"Giant Hula Hoops",1331
1234,"Dry Sherry, 1lt",5401

7. *Редактирование БД*. Добавьте функции для добавления/удаления товаров в БД. Функция **AddItem()** должна добавлять описание товара в БД. При этом если товар с таким штрихкодом уже существует в БД, он должен замещаться новым. Функция **RemoveItem()** должна удалять товар по его штрихкоду из БД.
8. *Анализ продаж*. Разработайте функцию **TotalSales()**, которая принимает на входе массив чеков и печатает на экране таблицу проданных товаров по всем чекам. В таблице должна присутствовать информация о названии товара, проданном количестве, сумме (возможно, с учетом скидки по соответствующим позициям).
9. Разработайте функцию **AnalyzeSales()**, которая принимает на входе массив чеков и печатает на экране таблицу пар товаров, которые чаще всего покупают вместе. Эта пара должна включаться в таблицу пар, если она присутствует более чем в одном чеке.

Летний блок индивидуальных заданий учащихся на летний период.

Рекомендуемые книги для прочтения:

1. Стивен Прата Язык программирования C++. Лекции и упражнения, 5-е издание / М.: Вильямс, 2007. – 1248 с.
2. Харви Дейтел, Пол Дейтел Как программировать на C++: Третье издание. Пер с англ. - М.: ЗАО "Издательство БИНОМ", 2003. – 1011 с.

Задачи

1. Златопольский, Д. М. Сборник задач по программированию / Д. М. Златопольский.– 2-е изд.– Санкт-Петербург: БХВ-Петербург, 2007.– 240 с.

3 задач: Глава 1. Ввод и вывод числовых данных. Оператор присваивания
2 задач: Глава 2. Целочисленная арифметика
5 задач: Глава 3. Величины логического типа
5 задач: Глава 4. Условный оператор
5 задач: Глава 5. Оператор цикла с параметром
5 задач: Глава 6. Операторы цикла с условием
5 задач: Глава 7. Сочетание оператора цикла и условного оператора
5 задач: Глава 8. Вложенные циклы
5 задач: Глава 9. Строки символов
Глава 10. Функции и процедуры
10 задач: Функции
10 задач: Процедуры
3 задач: Рекурсия
7 задач: Глава 11. Одномерные массивы
7 задач: Глава 12. Двумерные массивы
6 задач: Глава 13. Массивы величин типа “запись”
5 задач: Глава 14. Типизированные файлы
5 задач: Глава 15. Текстовые файлы
5 задач: Глава 16. Случайные числа