Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Волгоградский государственный технический университет
Факультет автоматизированных систем и технологической информатики
Кафедра «Автоматизация производственных процессов»

ПРОГРАММА ДИСЦИПЛИНЫ

Языки программирования систем реального времени

Направление подготовки
220400.62 «Управление в технических системах»
Профиль подготовки «Системы и технические средства
автоматизации и управления»

Факультет подготовки инженерных кадров
Заочная форма обучения
(сокращенная форма)

Курс	3
Семестр	6
Число зачетных единиц	3,0
Всего по учебному плану	108
Всего часов аудиторных занятий, час	10
Лекции, час	4
Лабораторные занятия, час	6
Форма итогового контроля	Зачет

Разработал профессор Яковлев А.А. e-mail: app@vstu.ru

Зав. кафедрой АПП Сердобинцев Ю.П.

Волгоград 2014

1. АННОТАЦИЯ ДИСЦИПЛИНЫ

«Языки программирования систем реального времени» является специальной дисциплиной, которая позволяет решать инженерные задачи, возникающие в процессе автоматизации промышленного производства. К их числу, в частности, относятся задачи автоматизации на основе микропроцессорных систем управления.

Дисциплина «Языки программирования систем реального времени» ставит своей целью изучение принципов функционирования программных средств в ЭВМ, архитектуры и функциональной организации операционных систем реального времени, а также организации взаимодействия операционных систем с прикладным программным обеспечением, использующих системные ресурсы вычислительных систем.

В процессе изучения дисциплины студент должен освоить основы проектирования многоцелевых систем управления технологическим оборудованием на микропроцессорной элементной базе, типовые аппаратные и программные решения, реализованные в серийных отечественных и зарубежных системах управления, архитектуру и программирование систем управления; научиться применять на практике весь комплекс знаний, полученный при изучении общеинженерных и специальных дисциплин, а также научиться читать, разрабатывать и описывать функциональные, структурные, принципиальные, общие электрические схемы и алгоритмы работы систем управления технологическими процессами; разрабатывать программное обеспечение для сбора информации и управления внешними устройствами.

Изучение дисциплины «Языки программирования систем реального времени» базируется на знании общетеоретических и общеинженерных дисциплин, таких, как «Информационные технологии», «Инженерная и компьютерная графика», «Электроника» и др. Основные положения дисциплины используются при изучении следующих курсов: «Вычислительные машины системы и сети», «Технические средства автоматизации и управления», «Микропроцессорные системы автоматизации и управления», а также «Программирование микропроцессорных систем».

2. СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

Таблица 2.1

№ темы	Название основных тем и вопросов, изучаемых в рамках дисциплины	Кол-во часов, отводимых на лекции
1	2	3
1	Введение. Цели и задачи дисциплины. Базовые средства языка С++. Состав языка, алфавит, идентификаторы, ключевые слова, знаки операций. Типы данных С++. Концепция типов данных. Основные и составные типы языка.	0,5
2	Структура программы на С++. Переменные и выражения. Базовые конструкции структурного программирования. Условные и циклические операторы. Массивы и указатели. Строки.	0,5
3	Типы данных, определяемые пользователем. Переименования типов, перечисления, структуры, битовые поля, объединения.	1,0
4	Модульное программирование. Функции и их параметры, передача массивов и имен функций в качестве параметров. Параметры со значениями по умолчанию, функции с переменным числом параметров. Рекурсивные функции, перегрузка функций. Шаблоны функций. Функции стандартной библиотеки. Директивы препроцессора.	0,5

Продолжение таблицы 2.1

1	2	3
5	Динамические структуры данных. Линейные списки, стеки, очереди, бинарные деревья. Реализация динамических структур с помощью массивов.	0,5
6	Классификация операционных систем. Управление процессами в системах реального времени. Понятие Процесса. Состояния процесса. Планирование процессов. Асинхронные параллельные процессы. Взаимодействие процессов. Пользовательский уровень. Синхрониизация процессов.	0,5
7	Управление внутренней памятью. Иерархическая организация памяти. Трансляция и загрузка модулей в абсолютных адресах. Файловая система. Копирование и восстановление информации. Производительность систем реального времени.	0,5
	ИТОГО	4

3.1. Лабораторные занятия

Таблица 3.1

Номер лаб. работы	Наименование лабораторной работы и вырабатываемые компетенции	Объем, час
1	Разветвляющиеся программы и циклы на С++. *Компетенции:* ОК-11; ПК-3; ПК-11; ПК-31	2
2	Работа с одномерными массивами и указателями. *Компетенции:* ПК-3; ПК-11	2
3	Работа со строками и файлами на языке С++ *Компетенции:* ОК-11; ПК-10; ПК-11	2
ИТОГО		6

4 Самостоятельная работа студентов

4.1 Цель работы

В течение шестого семестра студенты выполняют контрольную работу, в которой требуется разработать программу на языке *С*.

4.2. Задание

Напишите программу для расчета по двум формулам. Предварительно подготовьте тестовые примеры по второй формуле с помощью калькулятора (результат вычисления по первой формуле должен совпадать со второй). Список математических функций библиотеки *С++* приведен в приложении. Для их использования необходимо подключить к программе заголовочный файл `<math.h>`.

4.3. Варианты заданий контрольной работы

Варианты заданий контрольной работы (20 вариантов заданий) выбираются по двум последним цифрам зачетной книжки (например, вариант 02 при последних цифрах 02, 22, 42, 62, 82, вариант 12 при цифрах 12, 32, 52, 72, 92) и т. д.

Вариант 1

$$z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha)$$

$$z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right)$$

Вариант 2

$$z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$$

$$z_2 = 2\sqrt{2}\cos\alpha\sin\left(\frac{\pi}{4} + 2\alpha\right)$$

Вариант 3

$$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$$

$$z_2 = 2\sin\alpha$$

Вариант 4

$$z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha}$$

$$z_2 = tg3\alpha$$

Вариант 5

$$z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha$$

$$z_2 = \cos^2\alpha + \cos^4\alpha$$

Вариант 6

$$z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$$

$$z_2 = 4\cos\frac{\alpha}{2}\cdot\cos\frac{5}{2}\alpha\cdot\cos 4\alpha$$

Вариант 7

$$z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right)$$

$$z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}$$

Вариант 8

$$z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1$$

$$z_2 = \sin(y + x)\cdot\sin(y - x)$$

Вариант 9

$$z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$$

$$z_2 = -4\sin^2\frac{\alpha-\beta}{2}\cdot\cos(\alpha+\beta)$$

Вариант 10

$$z_1 = \frac{\sin\left(\frac{\pi}{2}+3\alpha\right)}{1-\sin(3\alpha-\pi)}$$

$$z_2 = ctg\left(\frac{5}{4}\pi+\frac{3}{2}\alpha\right)$$

Вариант 11

$$z_1 = \frac{1-2\sin^2\alpha}{1+\sin 2\alpha}$$

$$z_2 = \frac{1-tg\alpha}{1+tg\alpha}$$

Вариант 12

$$z_1 = \frac{\sin 4\alpha}{1+\cos 4\alpha}\cdot\frac{\cos 2\alpha}{1+\cos 2\alpha}$$

$$z_2 = ctg\left(\frac{3}{2}\pi-\alpha\right)$$

Вариант 13

$$z_1 = \frac{\sin\alpha+\cos(2\beta-\alpha)}{\cos\alpha-\sin(2\beta-\alpha)}$$

$$z_2 = \frac{1+\sin 2\beta}{\cos 2\beta}$$

Вариант 14

$$z_1 = \frac{\cos\alpha+\sin\alpha}{\cos\alpha-\sin\alpha}$$

$$z_2 = tg2\alpha+\sec 2\alpha$$

Вариант 15

$$z_1 = \frac{\sqrt{2b+2\sqrt{b^2-4}}}{\sqrt{b^2-4}+b+2}$$

$$z_2 = \frac{1}{\sqrt{b+2}}$$

Вариант 16

$$z_1 = \frac{x^2 + 2x - 3 + (x+1)\sqrt{x^2 - 9}}{x^2 - 2x - 3 + (x-1)\sqrt{x^2 - 9}}$$

$$z_2 = \sqrt{\frac{x+3}{x-3}}$$

Вариант 17

$$z_1 = \frac{\sqrt{(3m+2)^2 - 24m}}{3\sqrt{m} - \frac{2}{\sqrt{m}}}$$

$$z_2 = -\sqrt{m}$$

Вариант 18

$$z_1 = \left(\frac{a+2}{\sqrt{2a}} - \frac{a}{\sqrt{2a}+2} + \frac{2}{a-\sqrt{2a}}\right) \cdot \frac{\sqrt{a} - \sqrt{2}}{a+2}$$

$$z_2 = \frac{1}{\sqrt{a} + \sqrt{2}}$$

Вариант 19

$$z_1 = \left(\frac{1+a+a^2}{2a+a^2} + 2 - \frac{1-a+a^2}{2a-a^2}\right)^{-1} (5 - 2a^2)$$

$$z_2 = \frac{4 - a^2}{2}$$

Вариант 20

$$z_1 = \frac{(m-1)\sqrt{m} - (n-1)\sqrt{n}}{\sqrt{m^3 n} + nm + m^2 - m}$$

$$z_2 = \frac{\sqrt{m} - \sqrt{n}}{m}$$

Приложение. Математические функции (заголовочный файл <math.h>)

acos	Возвращает арккосинус аргумента
asin	Возвращает арксинус аргумента
atan	Возвращает арктангенс аргумента
atan2	Возвращает арктангенс отношения аргументов
ceil	Округляет вверх
cos	Вычисляет косинус
cosh	Вычисляет гиперболический косинус
exp	Возвращает степень числа e
fabs	Возвращает модуль числа
floor	Округляет вниз
fmod	Возвращает остаток от деления x на y
frexp	Выделяет из числа мантиссу и экспоненциальную часть
ldexp	Преобразует мантиссу и показатель степени в число
log	Вычисляет натуральный алгоритм
log10	Вычисляет логарифм по основанию 10
modf	Разбивает число на целую и дробную части
pow	Возводит число в степень
sin	Вычисляет синус
sinh	Вычисляет гиперболический синус
sqrt	Вычисляет квадратный корень
tan	Возвращает тангенс аргумента
tanh	Возвращает арктангенс аргумента

4.4 Общие методические указания

Программа на языке *C* состоит из функций, описаний и директив препроцессора. Препроцессором называется предварительная фаза компиляции. Одна из функций должна иметь имя *main*. Выполнение программы начинается с первого оператора этой функции. Простейшее определение функции имеет следующий формат:

```
тип_возвращаемого_значения имя ([ параметры ]){
   операторы, составляющие тело функции
}
```

Как правило, функция используется для вычисления какого-либо значения, поэтому перед именем функции указывается его тип. Функции в языке *C* имеют следующие особенности:

если функция не должна возвращать значение, указывается тип `void`.

тело функции является блоком и заключается в фигурные скобки;

функции не могут быть вложенными;

каждый оператор заканчивается точкой с запятой.

Пример структуры программы, содержащей функции `main`, `f1` и `f2`:

```
директивы препроцессора
описания
int main(){
   операторы главной функции
}
int f1(){
   операторы функции f1
}
int f2(){
   операторы функции f2
}
```

В языке *C* нет встроенных средств ввода/вывода – он осуществляется с помощью функций, типов и объектов, содержащихся в стандартных библиотеках. Используется два способа: функции, унаследованные из языка *C*, и объекты *C++*.

Основные функции ввода/вывода в стиле C выглядят следующим образом:

```
int scanf(const char* format,…) // ввод
int printf(const char* format,…) // вывод
```

Они выполняют форматированный ввод и вывод произвольного количества величин. Строка формата содержит символы, которые при выводе копируются на экран или запрашиваются с клавиатуры при вводе, и спецификации преобразования, начинающиеся со знака %, которые при вводе и выводе заменяются конкретными величинами.

Пример программы, использующей функции ввода/вывода в стиле *C*:

```
#include <stdio.h>
int main(){
   int i;
   printf("Введите целое число\n");
   scanf("%d", &i);
   printf("Вы ввели число %d, спасибо!", i);
   return 0;
}
```

Первая строка этой программы – директива препроцессора, по которой в текст программы вставляется заголовочный файл `<stdio.h>`, содержащий описание использованных в программе функций ввода/вывода (в данном случае угловые скобки являются элементом языка). Все директивы препроцессора начинаются со знака #. Третья строка – описание переменной целого типа с именем `i`.

Функция `printf` в четвертой строке выводит приглашение «Введите целое число» и переходит на новую строку в соответствии с управляющей последовательностью `\n`. Функция `scanf` заносит введенное с клавиатуры целое число в переменную `i` (знак `&` означает операцию получения адреса), а следующий оператор выводит на экран указанную в нем строку, заменив спецификацию преобразования на значение этого числа.

Ниже приведена та же программа с использованием библиотеки классов *C++*:

```
#include <iostream.h>
int main(){
   int i;
   cout << "Введите целое число\n";
   cin >> i;
   cout << "Вы ввели число " << i << ", спасибо!";
   return 0;
}
```

Заголовочный файл <iostream.h> содержит описание набора классов для управления вводом/выводом. В нем определены стандартные объекты-потоки `cin` для ввода с клавиатуры и `cout` для вывода на экран, а также операции помещения в поток << и чтения из потока >>.

В дальнейшем изложении будут использоваться оба способа, но в одной программе смешивать их не рекомендуется.

4.5. Практическая часть

Программа перевода температуры в градусах по Фаренгейту в градусы Цельсия

Температура в градусах по Фаренгейту переводится в градусы Цельсия по формуле:

$$C = \frac{5}{9}(F - 32),$$

где C – температура по Цельсию, а F – температура по Фаренгейту.

Перед написанием любой программы надо четко определить, что в нее требуется ввести и что нужно получить в результате. В качестве исходных данных выступает одно вещественное число, представляющее собой температуру по Фаренгейту, в качестве результата – другое вещественное число – температура по Цельсию. Напишите в редакторе среды программирования следующий код.

```
#include <iostream.h>
#include <stdlib.h>
int main(){                                                    // 1
   float fahr, cels;                                           // 2
   cout << endl << " Введите температуру по Фаренгейту" << endl;   // 3
   cin >> fahr;                                                // 4
   cels = 5 / 9 * (fahr - 32);                                 // 5
   cout << " По Фаренгейту: " << fahr <<
      ", в градусах Цельсия: " << cels << endl;                // 6
   system("PAUSE");                                            // 7
   return 0;                                                   // 8
}
```

Рассмотрим каждую строку программы отдельно.

В начале программы записаны директивы препроцессора, по которой к исходному тексту программы подключается заголовочные файлы <iostream.h> и <stdlib.h>. Это текстовые файлы, которые содержат описания элементов стандартных библиотек, необходимых, например, для выполнения ввода-вывода. Директивы препроцессора записываются в отдельной строке, перед знаком # могут находиться только пробельные символы.

Оператор 1 (строка, помеченная комментарием //1) представляет собой заголовок главной функции программы. Она должна иметь имя main, указывающее, что именно с нее требуется начинать выполнение. За именем функции в скобках следует список передаваемых ей параметров. В данном случае он пуст (скобки необходимы, чтобы компилятор мог распознать функцию). Перед именем записан тип значения (`int` – целое), возвращаемого функцией в точку ее вызова. Главная функция должна возвращать целочисленное значение. После круглых скобок в фигурных скобках записывается тело функции. Для удобства восприятия принято располагать тело с отступом в 3–4 позиции от заголовка.

Для хранения исходных данных и результатов необходимо выделить место в оперативной памяти (оператор 2). В данной программе требуется хранить два значения: температуру по Цельсию (`cels`) и температуру по Фаренгейту (`fahr`), поэтому в операторе определяются две переменные.

Поскольку температура может принимать не только целые значения, для переменных выбран вещественный тип `float`. Можно также выбрать тип `double`, позволяющий представлять вещественные числа большего диапазона значений и с большей точностью.

Для того, чтобы пользователь программы знал, в какой момент требуется ввести с клавиатуры данные, применяется так называемое приглашение к вводу (оператор 3). На экран выводится указанная в операторе строка символов, и курсор переводится на следующую строку.

Стандартный объект, с помощью которого выполняется вывод на экран, называется `cout`. Ему с помощью операции << передается то, что нужно вывести. Для вывода нескольких элементов используется цепочка таких операций. Для перехода на следующую строку записывается так называемый манипулятор `endl`, который управляет, то есть «манипулирует» стандартным объектом `cout`.

В операторе 4 выполняется ввод с клавиатуры одного числа в переменную `fahr`. Для этого используется стандартный объект `cin` и операция извлечения (чтения) >>. Если требуется ввести несколько величин, используется цепочка операций >>. В процессе ввода число преобразуется из последовательности символов, набранных на клавиатуре, во внутреннее

представление вещественного числа и помещается в ячейку памяти, зарезервированную для переменной fahr.

В операторе 5 вычисляется выражение, записанное справа от операции присваивания (=), и результат присваивается переменной cels.

Порядок вычислений определяется приоритетом операций. В случае сомнений надо обращаться к справочной информации. Основные правила соответствуют принятым в математике: вычитание имеет более низкий приоритет, чем умножение, поэтому для того, чтобы оно было выполнено раньше, соответствующая часть выражения заключается в скобки. Деление и умножение имеют одинаковый приоритет и выполняются слева направо.

Для вывода результата в операторе 6 применяется объект cout. Выводится цепочка, состоящая из четырех элементов. Это строка " По Фаренгейту: ", значение переменной fahr, строка ", в градусах Цельсия:" и значение переменной cels. При выводе строк все символы, находящиеся внутри кавычек, включая и пробелы, выводятся без изменений. При выводе значений переменных выполняется преобразование из внутреннего представления числа в строку символов, представляющую это число. Под значение отводится ровно столько позиций, сколько необходимо для вывода всех его значащих цифр. Это значит, что если вывести две переменные подряд, их значения «склеятся». Поэтому рекомендуется всегда предварять выводимые значения текстовыми пояснениями.

Оператор 7 необходим для прерывания выполнения программы после вывода на экран результатов вычисления. Вызывается функция system, которая передает строку командному процессору операционной системы. Для ее вызова в программу включен заголовочный файл <stdlib.h>.

Последний оператор (8) программы предназначен для возврата из нее и передачи значения во внешнюю среду. В главной функции разрешается его опускать, но в этом случае компилятор может выдать предупреждение.

Наберите текст программы, скомпилируйте ее и запустите.

Вам приготовлен неприятный сюрприз: вместо приглашения «Введите температуру по Фаренгейту» вы увидите набор странных символов. Это объясняется тем, что в операционной системе *MS DOS* для кодировки символов используется стандарт *ASCII*, являющийся международным только в первой половине кодов (от 0 до 127), вторая половина кодов (от 128 до 255) является национальной и различна для разных стран.

Например, в России для второй половины таблицы *ASCII* используется так называемая «альтернативная кодировка ГОСТа». В *Windows* же используется стандарт *ANSI*, в первой половине совпадающий с *ASCII*, а во второй половине отличающийся от его российского варианта. Режим консольных приложений должен имитировать работу в среде MS DOS, поэтому ввод-вывод выполняется в этом режиме в кодировке *ASCII*. В то же время в текстовом редакторе среды программирования используется кодировка *ANSI*.

Для выхода из этого положения в программе будет использоваться функция CharToOem() для преобразования символов из кодировки *ANSI* в кодировку *ASCII*. Для обратного преобразования можно использовать функцию OemToChar(). Первая из названных функций нужна для вывода русскоязычного текста на экран, а вторая – для ввода такого текста с клавиатуры, если в дальнейшем потребуется запись этого текста в документы (файлы) с кодировкой *ANSII*. Чтобы использовать обе эти функции, нужно подключить к программе заголовочный файл <windows.h>.

```
#include <iostream.h>
#include <stdlib.h>
#include <windows.h>
char* Rus(const char* text);                                    // 0

int main(){                                                     // 1
   float fahr, cels;                                            // 2
```

```
    cout << endl << Rus("Введите температуру по Фаренгейту") << endl;   // 3
    cin >> fahr;                                                        // 4
    cels = 5 / 9 * (fahr - 32);                                         // 5
    cout << Rus("По Фаренгейту: ") << fahr;                             // 6
    cout << Rus(", в градусах Цельсия: ") << cels << endl;              // 7
    system("PAUSE");                                                    // 8
    return 0;                                                           // 9
}
char bufRus[256];                                                       // 10
char* Rus(const char* text){                                            // 11
    CharToOem(text, bufRus);                                            // 12
    return bufRus;                                                      // 13
}
```

Для решения проблемы в программу добавлена функция `Rus()`, которая, обращаясь к функции `CharToOem()`, возвращает в качестве адреса преобразованной строки указатель на char (строки 11–13). В качестве временного хранилища преобразуемой строки функция использует глобальный массив `bufRus` длиной 256 символов, описанный в операторе 10. Любая строковая_константа в программе заменяется выражением `Rus(строковая_константа)`. То, что функция `Rus()` возвращает значение типа `char*`, приводит к следующему ограничению: ее нельзя использовать более одного раза в цепочке операций << для одного объекта `cout`.

Запустите программу на выполнение несколько раз, задавая различные значения температуры. Не забудьте, что число, имеющее дробную часть, при вводе следует записывать с точкой, а не с запятой. Можно задавать и целые числа – они будут автоматически преобразованы в вещественную форму.

Как вы можете видеть, результат выполнения программы оказывается равным нулю! Это происходит из-за способа вычисления выражения. Константы 5 и 9 (в операторе 5) имеют целый тип, поэтому результат их деления также целочисленный. Округления при этом не происходит, дробная часть всегда отбрасывается. Чтобы исправить эту ошибку – достаточно записать хотя бы одну из констант в виде вещественного числа, например:

```
cels = 5. / 9 * (fahr - 32);                                            // 5
```

Вещественная константа «5.» по умолчанию имеет тип `double`, и при выполнении деления происходит автоматическое преобразование к этому же типу другой константы, а затем и результата вычитания.

Можно записать это выражение и по-другому, просто изменив порядок действий:

```
cels = 5 * (fahr - 32) / 9;                                             // 5
```

В этом случае будет выполнено преобразование констант к типу `float`, как к наиболее длинному из участвующих в выражении.

Теперь программа будет работать верно и выдаст следующее:

```
Введите температуру по Фаренгейту
451
По Фаренгейту: 451, в градусах Цельсия: 232.778
```

Попробуйте при вводе температуры задать нецифровые символы и проинтерпретировать полученный результат.

Как видите, даже в простом примере можно допустить ошибки! В данном случае заметить их легко, но так происходит далеко не всегда, поэтому запомните важное правило: надо всегда заранее знать, что должна выдать программа. Добиться этого можно разными способами, например, вычислением результатов в уме, на бумаге или с помощью калькулятора, а в более сложных случаях – расчетами по альтернативной или упрощенной методике.

Напишите ту же программу вторым способом, с использованием функций библиотеки *C++*, унаследованных из языка *C*. Этот способ применяется достаточно часто, потому что в использовании этих функций есть свои преимущества.

К программе подключается другой заголовочный файл – <stdio.h>. Он содержит описание функций, констант и других элементов, относящихся ко вводу-выводу «в стиле *C*».

Ниже приведен текст программы для работы под управлением операционной системы *MS DOS*. Самостоятельно добавьте необходимый код для вывода на экран кириллицы и приостановки программы после вывода результатов вычисления.

```
#include <stdio.h>
int main(){                                                    // 1
   float fahr, cels;                                           // 2
   printf("\n Введите температуру по Фаренгейту\n");           // 3
   scanf("%f", &fahr);                                         // 4
   cels = 5 * (fahr - 32) /9;                                  // 5
   printf("По Фаренгейту: %6.2f, в градусах Цельсия: %6.2f\n",
        fahr, cels);                                           // 6
   return 0;                                                   // 7
}
```

Рассмотрим отличия этой программы от предыдущей.

Функция `printf` в операторе 3 выполняет вывод переданного ей в качестве параметра строкового литерала, то есть последовательности любых символов в кавычках, на стандартное устройство вывода (дисплей). Символы `\n` называются управляющей последовательностью. Есть разные управляющие последовательности. Эта последовательность задает переход на следующую строку.

Для ввода исходных данных в операторе 4 используется функция `scanf`. В ней требуется указать формат вводимых значений, а также адреса переменных, которым они будут присвоены. В первом параметре функции `scanf` в виде строкового литерала задается спецификация формата вводимой величины, соответствующая типу переменной. Спецификация `%f` соответствует типу `float`. В качестве второго параметра функции передается адрес переменной, по которому будет помещено вводимое значение. Операция взятия адреса обозначается `&`.

Для вывода результата в операторе 6 применяется функция `printf`. Первый параметр, имеющий вид строкового литерала, задает вид и формат выводимой информации. Второй и третий параметры представляют собой имена переменных. Все символы литерала, кроме спецификаций формата `%f` и управляющей последовательности `\n`, выводятся на дисплей без изменений. При выводе форматные спецификации будут заменены конкретными значениями переменных `fahr` и `cels`.

Формат вывода чисел можно уточнить при помощи так называемых модификаторов формата – чисел, которые записаны перед спецификацией. Первое число задает минимальное количество позиций, отводимых под выводимую величину, второе – сколько из этих позиций отводится под дробную часть величины. Необходимо учитывать, что десятичная точка тоже занимает одну позицию. Если заданного количества позиций окажется недостаточно для размещения числа, компилятор автоматически выделит поле достаточной длины.

Часто используются еще две спецификации: `%d` – для величин целого типа в десятичной системе счисления, `%lf` – для величин типа `double`.

Для каждой программы необходимо выбрать один наиболее удобный способ вывода (либо с помощью функций, либо с помощью классов), поскольку смешивать оба варианта не рекомендуется.

Используйте функции ввода/вывода в тех программах, где требуется тщательное форматирование результатов, а классы – в остальных случаях.

Программа расчета временного интервала

Требуется для заданных моментов начала и конца некоторого промежутка времени в часах, минутах и секундах (в пределах одних суток) найти продолжительность этого промежутка в тех же единицах.

Исходными данными для этой задачи являются шесть целых величин, задающих моменты начала и конца интервала, результатами – три целых величины.

Тип переменной выбирается, исходя из диапазона и требуемой точности представления данных, а имя дается в соответствии с ее содержимым. Так как потребуется хранить исходные данные, не превышающие величины 60 для минут и секунд и величины 24 для часов, можно ограничиться коротким целым типом (`short int`, сокращенно `short`). Переменные для хранения начала интервала обозначаются `hour1`, `min1` и `sec1`, для хранения конца интервала – `hour2`, `min2` и `sec2`, а результирующие величины – `hour`, `min` и `sec`.

Для решения этой задачи необходимо преобразовать оба момента времени в секунды, вычесть первый из второго, а затем преобразовать результат обратно в часы, минуты и секунды. Следовательно, потребуется промежуточная переменная, в которой будет храниться интервал в секундах. Она может иметь весьма большие значения (в сутках 86400 с). В величинах типа `short` могут храниться значения, не превышающие 65535 для величин без знака (`unsigned short`), поэтому следует выбрать длинный целый тип (`long int`, сокращенно `long`). Целый тип `int` в зависимости от архитектуры компьютера может совпадать либо с коротким, либо с длинным целым типом. Ниже приведен текст программы:

```
#include <iostream.h>
int main(){
   short hour1, min1, sec1, hour2, min2, sec2, hour, min, sec;
   cout << endl << "Введите время начала интервала (час мин сек)" << endl;
   cin >> hour1 >> min1 >> sec1;
   cout << endl << "Введите время конца интервала (час мин сек)" << endl;
   cin >> hour2 >> min2 >> sec2;
   long sum_sec = (hour2 - hour1)*3600 + (min2 - min1)*60 + sec2 - sec1;
   hour = sum_sec / 3600;
   min = (sum_sec - hour*3600)/60;
   sec = sum_sec - hour*3600 - min*60;
   cout << "Продолжительность промежутка от " <<
      hour1 << ':' << min1 << ':' << sec1 << " до " <<
      hour2 << ':' << min2 << ':' << sec2 << endl << " равна " <<
      hour  << ':' << min  << ':' << sec  << endl;
   return 0;
}
```

Для перевода результата из секунд обратно в часы и минуты используется эффект отбрасывания дробной части при делении целого числа на целое.

ВНИМАНИЕ. Данные при вводе должны разделяться пробелами, символами перевода строки или табуляции (но не запятыми!).

Протестируйте программу на различных наборах исходных данных. Ниже приводится текст программы с использованием функций ввода-вывода в стиле C:

```
#include <stdio.h>
int main(){
   short hour1, min1, sec1, hour2, min2, sec2, hour, min, sec;
   printf("Введите время начала интервала (час мин сек)\n");
   scanf("%i%i%i", &hour1, &min1, &sec1);
   printf("Введите время конца интервала (час мин сек)\n");
   scanf("%i%i%i", &hour2, &min2, &sec2);
   long sum_sec = (hour2 - hour1)*3600 + (min2 - min1)*60 + sec2 - sec1;
```

```
    hour = sum_sec / 3600;
    min = (sum_sec - hour * 3600) / 60;
    sec = sum_sec - hour * 3600 - min * 60;
    printf("Длительность промежутка от %2i: %2i: %2i до %2i: %2i: %2i\n",
       hour1, min1, sec1, hour2, min2, sec2);
    printf(" равна %2i: %2i: %2i\n", hour, min, sec);
    return 0;
}
```

4.6 Наиболее важные моменты, которые следует запомнить

1. Выбирайте тип переменных с учетом диапазона и требуемой точности представления данных.

2. Давайте переменным имена, отражающие их назначение.

3. Ввод с клавиатуры предваряйте приглашением. Для контроля сразу же после ввода выводите исходные данные на дисплей (по крайней мере, в процессе отладки).

4. До запуска программы подготовьте тестовые примеры, содержащие исходные данные и ожидаемые результаты. Отдельно проверьте реакцию программы на неверные исходные данные.

5. При записи выражений обращайте внимание на приоритет операций.

6. В функциях `printf` и `scanf` для каждой переменной указывайте спецификацию формата, соответствующую ее типу. Не забывайте, что в `scanf` передается адрес переменной, а не ее значение.

7. При использовании стандартных функций или классов требуется с помощью директивы `#include` подключить к программе соответствующие заголовочные файлы. Установить, какой именно файл необходим, можно с помощью справочной системы.

8. Не смешивайте в одной программе ввод/вывод с помощью классов (в стиле *C*++) и с помощью функций библиотеки (в стиле *C*).

9. Отдавайте предпочтение локальным переменным перед глобальными. Переменная должна иметь минимальную из возможных областей действия.

10. Данные при вводе разделяйте пробелами, символами перевода строки или табуляции.

5 Содержание контрольной работы

Контрольная работа состоит из двух частей – теоретической части и практической. В первой части необходимо письменно ответить на вопрос к зачету (см. п. 6), номер которого соответствует варианту, выбранному согласно п. 4.3. Во второй части контрольной работы необходимо представить полный текст написанной согласно варианту задания программы с соответствующими комментариями. Для разработки программы можно использовать любую среду программирования, поддерживающую язык *C* или *C*++.

6. Вопросы к зачету по дисциплине

1. Состав и алфавит языка *C*.
2. Идентификаторы, ключевые слова и знаки операций в языке *C*++. Комментарии.
3. Константы в языке *C*++, их типы и форматы. Управляющие последовательности.
4. Концепция типа данных. Основные типы данных в *C*++. Целый тип данных.
5. Символьный, расширенный символьный, логический типы данных, тип void, и типы данных с плавающей точкой в языке *C*++.
6. Структура программы в *C*++. Ввод и вывод данных.
7. Переменные в *C*++, формат их описания. Область действия и видимости идентификаторов.

8. Унарные операции в *C*++. Операции инкремента, декремента, отрицания и сдвига.
9. Бинарные операции в *C*++. Операции деления и остатка от деления, отношения и логические операции.
10. Операции присваивания и условная операция в языке *C*++.
11. Выражения в языке *C*++. Состав выражений, определение приоритета операций, результат и преобразование типов в выражениях.
12. Базовые конструкции структурного программирования. Оператор выражение.
13. Условные операторы в языке *C*++.
14. Операторы цикла, их структура и виды. Цикл *«while»* в языке *C*++.
15. Циклы «*do while*» и «*for*» в языке *C*++.
16. Операторы передачи управления в языке *C*++.
17. Указатели в языке *C*++ и их виды.
18. Инициализация указателей в языке *C*++.
19. Операция разадресации в языке *C*++.
20. Арифметические операции с указателями.
21. Ссылки в языке *C*++. Передача параметров в функции по ссылке.
22. Статические массивы в языке *C*++. Инициализация массива и доступ к его элементам.
23. Динамические массивы в языке *C*++. Способы выделения и освобождения памяти.
24. Строки в языке *C*++. Использование указателей при работе со строками.
25. Переименование типов и перечисления в языке *C*++.
26. Структуры в языке *C*++. Инициализация и способы доступа к полям.
27. Битовые поля и объединения в языке *C*++.
28. Функции в языке *C*++, их объявление и определение.
29. Обмен информацией в программе на языке *C*++ с помощью глобальных переменных и возвращаемых значений.
30. Обмен информацией в программе на языке *C*++ с помощью параметров. Виды передаваемых параметров.

7 Список основной и дополнительной литературы

Основная литература

1. *Камаев В. А*. Технологии программирования: учебник / В. А. Камаев, В. В. Костерин. – М.: Высш. Шк., 2006. – 359 с.

2. *Павловская Т. А*. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – СПб.: Питер, 2004. – 461 с.: ил.

3. *Павловская Т. А*. C/C++. Структурное программирование: Практикум / Т. А. Павловская, Ю. А. Щупак – СПб.: Питер, 2003. – 240 с.: ил.

4. *Павловская Т. А*. C/C++. Объектно-ориентированное программирование: Практикум / Т. А. Павловская, Ю. А. Щупак – СПб.: Питер, 2004. – 240 с.

5. *Яковлев А. А*. Программирование и основы алгоритмизации / А. А. Яковлев – Волгоград. гос. техн. ун-т. – Волгоград, 2004. – 96 с.

6. *Борзунова Т. Л*. Введение в динамическое программирование / Т. Л Борзунова, М. П. Барыкин М. П. – Волгоград. гос. техн. ун-т. – Волгоград, 2007. – 96 с.

Дополнительная литература

7. *Себеста, Р*. Основные концепции языков программирования / Р. Себе-ста; 5-е изд.: пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 672 с.

8. *Кормен Т*. Алгоритмы: построение и анализ / Кормен Т., Лейзерсон Ч., Ривест Р. М.: МЦНМО, 2000. – 960 с., 263 ил.

9. *Румянцев Д*. Путь программиста: опыт созидания личности программиста / Румянцев Д., Монастырский Л. М.: «Издательский дом ИНФРА–М», 2000. – 864 с., ил.

10. *Конолли Т.* Базы данных: проектирование, реализация и сопровождение. Теория и практика / Конолли Т., Бегг К., Страчан А. 2-е изд.: Пер с англ.: Уч. пос. – М.: Издательский дом «Вильямс», 2000. – 1120 с.

11. *Дал У.* Структурное программирование./ Дал У., Дейкстра Э., Хоор К. Пер. С англ. – М.: Мир, 1975.–247 с.

12. *Мандел, Т.* Разработка пользовательского интерфейса / Т. Мандел; пер. с англ. – М.: ДМК Пресс, 2001. – 416 с.

8 Перечень методических указаний

1. Разработка линейных программ на С++: метод. указания / сост. А. А. Яковлев / ВолгГТУ. – Волгоград, 2008. – 16 с.

2. Разветвляющиеся программы и циклы на С++: метод. указания / сост. А. А. Яковлев / ВолгГТУ. – Волгоград, 2008. – 16 с.

3. Работа с одномерными массивами и указателями: метод. указания / сост. А. А. Яковлев / ВолгГТУ. – Волгоград, 2009. – 16 с.

4. Двумерные массивы: метод. указания / сост. А. А. Яковлев / ВолгГТУ. – Волгоград, 2011. – 16 с.

5. Работа со строками и файлами на языке С++: / сост. А. А. Яковлев / ВолгГТУ. – Волгоград, 2011. – 16 с.