МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Институт сферы обслуживания и предпринимательства (филиал)
федерального государственного бюджетного образовательного
учреждения высшего образования «Донской государственный
технический университет» в г. Шахты Ростовской области
(ИСОиП (филиал) ДГТУ в г. Шахты)

КОЛЛЕДЖ ЭКОНОМИКИ И СЕРВИСА

На правах рукописи

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ

Методические указания
по выполнению практических работ
для подготовки обучающихся специальности
09.02.03 «Программирование в компьютерных системах»
очной формы обучения

Рассмотрены и рекомендованы для
использования в учебном процессе на
заседании педагогического совета
Протокол № 1от «31» сентября 2018 г

Шахты
ИСОиП (филиал) ДГТУ в г. Шахты
2019

Составитель:

Преподаватель
КЭС ИСОиП (филиал) ДГТУ в г. Шахты _____ В.В. Дубовсков
«___» _____ 2018 г.

Рецензенты:

Преподаватель первой категории
КЭС ИСОиП (филиал) ДГТУ в г. Шахты _____ И.А. Топоркова
«___» _____ 2018 г.

Преподаватель высшей категории
КЭС ИСОиП (филиал) ДГТУ в г. Шахты _____Л.В. Завгородняя
«___» _____ 2018 г.

Системное программирование: метод. указания по выполнению практических работ для подгот. обучающ. спец. 09.02.03 «Программирование в компьютерных системах» оч. и и заоч. форм обучения / сост. преп В.В. Дубовсков : Шахты, 2019. – 34с.

Настоящие методические указания определяют цели и задачи, содержание работ, общие требования к выполнению практических работ, форму отчетов, краткие теоретические сведения.

Данные методические указания предназначены для углубления и закрепления теоретических знаний, полученных обучающимися на уроках теоретического обучения, а также приобретения навыков самостоятельной работы по дисциплине Системное программирование.

Предназначено для обучающихся специальности 09.02.03 «Программирование в компьютерных системах».

Режим доступа к электронной копии печатного издания: http://www.libdb.sssu.ru

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Практикум по дисциплине «Системное программирование» предназначен для обучающихся специальности 09.02.03 «Программирование в компьютерных системах».

Методические указания по выполнению практических заданий разработаны в соответствии с требованиями Федерального государственного образовательного стандарта по специальности среднего профессионального образования 09.02.03 «Программирование в компьютерных системах» с учетом соответствующей учебной основной образовательной программы.

Целью практикума является закрепление теоретического материала, излагаемого в лекционном курсе. Практикум направлен на приобретение и развитие навыков самостоятельной работы по решению задач в области информатики.

Практикум состоит из практических работ и обеспечивает более глубокое изучение теоретического курса 5 семестра дисциплины. Для выполнения практических работ необходимы программные среды: ОС Windows, специализированные среды разработки программного обеспечения, офисное программное обеспечение (текстовый процессор, табличный процессор).

Задания и вопросы методических указаний соответствуют уровню подготовленности студентов к изучению данной дисциплины.

В методических указаниях определены цели, требования к выполнению заданий, приведены контрольные вопросы для самоподготовки и рекомендованы литературные источники.

В процессе подготовки к выполнению практических работ студентам следует изучить основные сведения из теории с использованием рекомендуемой литературы.

1. ОБЩИЕ ПОЛОЖЕНИЯ

Практическое занятие - это занятие, проводимое под руководством преподавателя в учебной аудитории, направленное на углубление теоретических знаний и овладение определенными методами самостоятельной работы. В процессе таких занятий вырабатываются практические умения.

Перед практическим занятием следует изучить конспект лекции и рекомендованную преподавателем литературу, обращая внимание на практическое применение теории и на методику решения типовых ситуаций. На практическом занятии главное – уяснить связь решаемых ситуаций с теоретическими положениями.

Для ведения записей на практических занятиях обычно заводят журнал практических занятий. Логическая связь лекций и практических занятий заключается в том, что информация, полученная на лекции, в процессе самостоятельной работы на практическом занятии осмысливается и перерабатывается, при помощи преподавателя анализируется до мельчайших подробностей, после чего прочно усваивается.

Успешное освоение курса «Информатика» предполагает активное, творческое участие студента путем планомерной, повседневной работы, которая позволит:

Знать:

– способы разработки системного программного обеспечения с учетом аппаратно-программных особенностей вычислительной машины;

– особенности современных систем программирования и принципы разработки системного программного обеспечения.

Уметь:

− разрабатывать программы в ОС Windows с графическим пользовательским интерфейсом;

− разрабатывать программы, в операционной системе UNIX с использованием системных вызовов;

− разрабатывать многопоточные программы с синхронизацией данных;

− разрабатывать динамически подключаемые библиотеки;

− перехватывать вызовы к операционной системе;

Владеть:

– навыками разработки программ в ОС Linux, Windows;

– навыками разработки многопоточных программ с синхронизацией данных;

– навыками разработки динамически подключаемых библиотек;

– навыками перехвата вызовов к операционной системе.

Представленные, в данных методических указаниях, практические задания направлены на формирование общих компетенций:

ОК-1: Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК-2: Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.
ОК-3: Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.
ОК-4: Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.
ОК-5: Использовать информационно-коммуникационные технологии в профессиональной деятельности.
ОК-6: Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.
ОК-7: Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.
ОК-8: Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.
ОК-9: Ориентироваться в условиях частой смены технологий в профессиональной деятельности.
ПК-1.1: Выполнять разработку спецификаций отдельных компонент.
ПК-1.2: Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.
ПК-1.3: Выполнять отладку программных модулей с использованием специализированных программных средств.
ПК-1.4: Выполнять тестирование программных модулей.
ПК-1.5: Осуществлять оптимизацию программного кода модуля.
ПК-1.6: Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

Наряду с формированием умений и навыков в процессе практических занятий обобщаются, систематизируются, углубляются и конкретизируются теоретические знания, вырабатывается способность и готовность использовать теоретические знания при решении задач.

При выполнении заданий студенты имеют возможность пользоваться лекционным материалом, с разрешения преподавателя осуществлять деловое общение с товарищами.

Оценка компетентности осуществляется следующим образом: по окончании выполнения задания студенты оформляют отчет, который затем выносится на завершающий этап формы изучения дисциплины. В процессе защиты выявляется информационная компетентность в соответствии с заданием на практическое занятие, затем преподавателем дается комплексная оценка деятельности студента.

Задачи:

- подтверждение теоретических положений;

- закрепление нового материала;
- взаимосвязь нового материала с пройденными темами;
- формирование исследовательских умений (наблюдать, сравнивать, анализировать, устанавливать зависимости, делать выводы и обобщения, самостоятельно вести исследование, оформлять результаты);
- обучение навыкам работы с текстом (понимать текст, различать его виды, анализировать содержащуюся в тексте информацию, делать выводы, различать точки зрения);
- формирование навыков работы в группе;
- обучение формулированию и аргументации своего мнения.

Требования к оформлению практических работ:

- цель работы;
- оснащение (оборудование, материалы и др.);
- теоретическая часть;
- практическая часть (порядок выполнения);
- выводы по работе;
- источники (литература);
- форма отчета практической работы (приказ № 227, раздел 5).

Пример оформления практической работы показан в Приложении А.

Критерии оценки выполненной работы:

- процент выполнения работы;
- достижение заданного результата;
- правильность выполнения заданий;
- наличие всех элементов работы;
- время выполнения работы.

2. ТИПЫ, ОПЕРАЦИИ, ВЫРАЖЕНИЯ

2.1. Верно ли записаны константы, представляющие целочисленные значения? Для верно записанных констант определить их значение, тип.

123 1E6 123456789LU -5 0XFUL
‘0’ 058 ‘\x7’ 0X-1AD ‘\122’
001230xffffffL 01A -‘x’ ″x″
‘a’U 0731UL ‘\n’ +0xaf 0X0

2.2. Верно ли записаны константы с плавающей точкой? Для верно записанных констант определить их значение, тип.

1.71 1E-6 0.314159E1F .005 0051E-04
5.E+20e0 0x1A1.5 05.5 0
0X1E6 0F 1234.56789L 1.0E-10D 3.1415U
1e-2f -12.3E-6 +10e6 123456L E-6

2.3. Верно ли записаны выражения? Для верно записанных выражений вычислить их значения (операции + - * / % =):

```
int a, b, c, d, e;
a = 2;  b = 13;  c = 7;  d = 19;  e = -4;
```

b / a / c d / a % c c % d-e -e % a + b / a ∗-5+5
b % e 7-d%+(3-a) b % - e ∗ c 9 / c - - 20 / d

2.4. Верно ли решена задача: «значение целочисленной переменной c увеличить на 1; целочисленной переменной a присвоить значение, равное удвоенному значению переменной c ».

```
int a, c;  c = 5;
```

a). c ++ ; b). a = 2 ∗ c++ ; c). c += 1; d). a = c++ + c;
a = 2 ∗ c; a = c + c;

e). ++c; f). a = ++ c + c; g). a = c += 1 + c; h). a = (c+=1)+c;
a = c + c;

2.5. Верно ли решена задача: «значение целочисленной переменной c уменьшить на 1; целочисленной переменной a присвоить значение, равное частному от деления переменной c на 2».

```
int a, c;  c = 5;
```

a). -- c ; b). a = -- c / 2; c). c -= 1; d). a = c -- / 2;
a = c / 2; a = c % 2;
e). a = c -= 1/2; f). a = (c = c - 1)/2; g). a = (c -= 1)/2; h).
a=(c-= 1)/2.0;

2.6. Эквивалентны ли выражения?

a) E1 op= E2 и E1 = E1 op E2
b) E1 op= E2 и E1 = E1 op (E2)

Замечание: здесь E1, E2 - выражения допустимого в этом случае типа ; op - операция (одна из + - * / % >> << & ^ |).

2.7. Верно ли записаны выражения? Для верно записанных выражений вычислить их значения (операции + - * / ++ - - операции присваивания):

int a, b, c; a = 2; b = 6; c = 3;

- - - a	-- - a	b-- - a	a += a ++	++ b / a ++ * --c
a --- b	- a-- -b	a ++ = b	a = a ++	b++ / ++a * c --
- --a	a- --c	a ++ = a	++ a = b	a = (b + 1) ++

2.8. Верно ли записаны выражения? Для верно записанных выражений вычислить их значения, определить тип результата (операции + - * / % ++ операции отношения, операции присваивания):

int i, j, k, m; char c, d; i = 1; j = 2; k = -7; m = 0; c = 'w';

d = 'a'+1 < c	m = - i - 5 * j >= k+1	i + j++ + k = = -2*j
m = 3 < j < 5	m = 3 = = j < 5	m = = c = 'w'
m = c != 87	m = c = ! 87	m = ! c = 87
m = !c+87	! m = =c + 87	m ! = c + 87
k = = j - 9 = = i	k *= 3 + j	i + j = !k
i += ++ j + 3	k %= m = 1 + n / 2	1 + 3 * n += 7 / 5
1 + 3 * (n += 7) / 5	c + i < c - 'x'+10	i - k = = '0'+9 < 10

2.9. В логике справедливы утверждения:

not (not x) = x

x and true = x

Верны ли соответствующие утверждения для операций ! и && в Си? Ответ обосновать.

2.10. При любом вещественном y > 0 x < x + y математически верно. Верно ли подобное утверждение для выражения на Си?

2.11. Написать эквивалентное выражение, не содержащее операции !

! (a>b)	! (2*a == b+4)	! (a<b && c<d)
! (a<2 || a>5)	! (a<1 || b<2 && c<3)	

2.12. Пусть

char c; short s; int i; unsigned u; signed char sc;

float f; double d; long lng; unsigned short us; long double ld;

Определить тип выражений:

c - s / i	u * 3 - 3.0 * u - i	u - us * i	(sc + d) * ld
(5 * lng - 'a') * (s + u / 2)	(f + 3) / (2.5f - s * 3.14)		

2.13. Допустимо ли в Си? Если "да" - опишите семантику этих действий; если "нет" - объясните почему.

a). . . .

```
int i;
i = (1 || 2) % (1 | 2 );
printf ( " i = %d\n", i);
```

b). . . .

```
int a, b, m, n, z;
m = n = 5;
z = a = b = 0;
z--, ( a = b ) = z + ( m != n );
printf ("%d %d %d %d %d\n",
        a, b, m, n, z);
```

c). . . .

```
int  i = 1;
i = i << i | i;
printf ( ″ i = %d\n″, i);
```

d). . . .

```
double  x = 1.9; int a;
double  b = 3.7;
a = b += (1 && 2 || 3) != (int)x;
printf (″%f %d %f\n″, x, a, b);
```

e). . . .

```
int x;
x = 5; ++ x =10;
printf ("%d\n", x);
```

f). . . .

```
int i, x, y;  x = 5; y = 10; i = 15;
x = ( y = 0, i = 1);
printf("%d %d %d\n", i, x ,y);
( x = y == 0) , i=1;
printf("%d %d %d\n", i, x, y);
```

g). . . .

h). . . .

```
int x, y;              int x = 2, y, z;
x = 5; y = x && ++ x; x *= 3+2; x *=y=z= 4;
printf("%d%d\n",x,y);printf ("%d %d %d\n",
                                    x, y, z);
            x = y == z; x == ( y = z );
            printf ("%d %d %d\n", x, y, z);
```

i). . . .

j). . . .

```
int x = 2, y = 1, z = 0;   int x=03, y=02,z= 01;
y = x && y || z;   printf("%d\n", x | y & -z);
x = x || !y && z; printf("%d\n", x ^ y & -z);
z = x / ++x;         printf("%d\n", x & y && z);
printf("%d %d %d\n", x, y, z); printf("%d\n",
                                    x<<3);
```

k). . . .

l). . . .

```
int x,y,z;x = y=z =1;int x,y,z,i;x =y =z =1;
x += y += z;       i = ++x || ++y && ++z;
      printf("%d\n", x < y ? y++ : x++);
    printf("%d%d%d%d\n", x,y,z,i);
    printf("%d\n", z+=x<y ? ++x : y--);
    i = x++ <= --y || ++z >= i;
            printf("%d %d %d\n", x, y, z);
             printf("%d%d%d%d\n", x,y,z,i);
           printf("%d\n", z>=y && y>=x);
```

2.14. Что будет напечатано в результате выполнения следующего фрагмента программы?

```
double d; float f; long lng; int i; short s;
s = i = lng = f = d = 100/3;
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
d = f = lng = i = s =100/3;
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
s = i = lng = f = d = 1000000/3;
```

```
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
d = f = lng = i = s =1000000/3;
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
lng = s = f = i = d =100/3;
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
f = s = d = lng = i = (double)100/3;
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
s = i = lng = f = d = 100/(double)3;
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
f = s = d = lng = i = (double)100/3;
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
i = s = lng = d = f = (double)(100/3);
printf("s = %hd i = %d lng = %ld f = %f d = %f\n", s, i, lng, f, d);
```

2.15. Что будет напечатано в результате выполнения следующего фрагмента программы?

```
double d = 3.2, x; int i = 2, y;
x = ( y = d / i ) * 2; printf ("x = %f ;y = %d\n", x, y);
x = ( y = d / i ) * 2; printf ("x = %d ;y = %f\n", x, y);
y = ( x = d / i ) * 2; printf ("x = %f ;y = %d\n", x, y);
y = d * ( x = 2.5 / d); printf ("x = %f; y = %d\n", x, y);
x = d * ( y = ( (int)2.9 + 1.1) / d; printf ("x = %d y = %f\n", x, y);
```

2.16. Дано вещественное число x . Не пользуясь никакими операциями, кроме умножения, сложения и вычитания, вычислить

$2x^4-3x^3+4x^2-5x+6$.

Разрешается использовать не более четырех умножений и четырех сложений и вычитаний.

2.17. Целой переменной k присвоить значение, равное третьей от конца цифре в записи целого положительного числа x.

2.18. Целой переменной k присвоить значение, равное сумме цифр в записи целого положительного трехзначного числа x.

2.19. Целой переменной k присвоить значение, равное первой цифре дробной части в записи вещественного положительного числа x.

2.20. Определить число, полученное выписыванием в обратном порядке цифр заданного целого трехзначного числа.

2.21. Идет n-ая секунда суток. Определить, сколько полных часов и полных минут прошло к этому моменту.

2.22. Дано вещественное число x . Не пользуясь никакими операциями, кроме умножения, получить

a) x^{21} за шесть операций

b) x^3 и x^{10} за четыре операции

c) x^5 и x^{13} за пять операций

d) x^2, x^5 и x^{17} за шесть операций

e) x^4, x^{12} и x^{28} за шесть операций

2.23. Выражения, соединенные операциями && и || , по правилам Си вычисляются слева направо; вычисления прекращаются, как только становится известна истинность или ложность результата. В других языках программирования, например в Паскале, вычисляются все части выражения в любом случае. Приведите «за» и «против» каждого из этих решений.

2.24. Почему в Си не допускается, чтобы один и тот же литерал-перечислитель входил в два различных перечислимых типа? Могут ли совпадать имена литералов-перечислителей и имена обычных переменных в одной области видимости? Могут ли разные литералы-перечислители иметь одинаковые значения?

2.25. «Упаковать» четыре символа в беззнаковое целое. Длина беззнакового целого равна 4.

2.26. «Распаковать» беззнаковое целое число в четыре символа. Длина беззнакового целого равна 4.

2.27. Заменить в целочисленной переменной x n бит, начиная с позиции p, n старшими инвертированными битами целочисленной переменной y.

2.28. Циклически сдвинуть значение целочисленной величины на n позиций вправо.

2.29. Циклически сдвинуть значение целочисленной величины на n позиций влево.

2.30. Выясните некоторые свойства и особенности поведения доступного Вам транслятора Си:

a) выяснить, сколько байт отведено для хранения данных типа short, int, long, float, double и long double;

b) выяснить способ представления типа char (signed- или unsigned- вариант);

c) проконтролировать, все ли способы записи констант допустимы:

- целых (обычная форма записи, u/U, l/L, их комбинации; запись констант в восьмеричной и шестнадцатиричной системах счисления)

- вещественных (обычная форма записи, в экспоненциальном виде, f/F, l/L, e/E)

- символьных (обычная форма записи, с помощью эскейп-последовательности) и строковых (в частности, происходит ли конкатенация рядом расположенных строковых констант)

d) выяснить, как упорядочены коды символов '0' - '9', 'a' - 'z', 'A' - 'Z', пробел (между собой и относительно друг друга);

e) проконтролировать, происходит ли инициализация переменных по умолчанию;

f) проверить, реагирует ли транслятор на попытку изменить константу;

g) исследовать особенности выполнения операции % с отрицательными операндами;

h) проверьте, действительно ли операции отношения == и != имеют более низкий приоритет, чем все другие операции отношения;

i) проверьте, действительно ли выполняется правило "ленивых вычислений" выражений в Си, т.е. прекращается ли вычисление выражений с логическими операциями, если возможно "досрочно" установить значение результата;

j) проверьте, все ли виды операнда операции sizeof (X), определяемые стандартом для арифметических типов, допускаются компилятором; действительно ли выражение X не вычисляется.

3. СИНТАКСИС И СЕМАНТИКА ОПЕРАТОРОВ ЯЗЫКА СИ

3.1. Перечислить все ситуации, когда в программах на Си используется составной оператор.

3.2. В Си точка с запятой используется в качестве признака конца оператора; в Паскале - в качестве разделителя операторов. Сравните эти решения, сформулируйте возможные «за» и «против».

3.3. Эквивалентны ли следующие фрагменты программы:

```
if (e1) if (e2) S1; else S2;
if (e1) { if (e2) S1; else S2; }
if (e1) { if (e2) S1; } else S2;
if (e1)  if (e2) S1; else ; else S2;
if (e1)  if (e2) S1; else S2; else ;
```

Замечание: здесь e1 и e2 - выражения допустимого в этом случае типа; S1 и S2 - произвольные операторы.

3.4. Описать в виде блок-схемы семантику каждого оператора цикла в Си (с учетом операторов break и continue, которые, возможно, содержатся в теле цикла).

3.5. Может ли быть определено число итераций цикла for до начала его выполнения?

в Паскале

в Си

3.6. Верно ли решена задача: «найти сумму первых 100 натуральных чисел»?

```
a) i = 1; sum = 0;
   for ( ; i <= 100; i++) sum += i;
b) sum = 0;
   for ( i = 1; i <= 100;) sum += i++;
c) for ( i = 1, sum = 0; i <= 100; sum += i+,  i++);
d) for ( i = 1, sum = 0; i <= 100; sum += i++);
e) for ( i = 0, sum = 0; i++, i <= 100; sum += i);
```

3.7. Выразить семантику цикла for с помощью цикла while. Эквивалентны ли полученные фрагменты программ?

3.8. Эквивалентны ли следующие фрагменты программы:

```
for ( ; e2 ; ) S;        и        while ( e2 ) S;
for ( ; ; ) S;           и        while (1) S;
```

Замечание: здесь e2 - выражение допустимого в этом случае типа; S - произвольный оператор.

3.9. Можно ли написать фрагмент программы на Си, эквивалентный данному, используя один оператор цикла for с пустым оператором в качестве тела цикла?

```
i = 0; c = getchar();
while (c != ’  ’ && c != ’\n’ && c !=’ \t’ && c != EOF)
   { i++; c = getchar(); }
```

3.10. Сравнить семантику операторов repeat в Паскале и do-while в Си.

3.11. Улучшить стиль (структуру) следующих фрагментов программы на Си:

```
a) while ( E1 )
{ if ( E2 ) continue; S; }
b) do  { if ( E1 ) continue; else S1; S2; }
while ( E2 );
```

Замечание: здесь E1, E2 - выражения допустимого в этом случае типа; S, S1, S2 - произвольные операторы.

3.12. Что напечатает следующая программа?

```
# include <stdio.h>
main()
{ int x, y, z;
x = y = 0;
while ( y < 10 ) ++y; x += y;
printf ("x = %d y = %d\n", x, y);
x = y = 0;
while ( y < 10 ) x += ++ y;
printf (" x= %d y = %d\n", x, y);
y = 1;
while ( y < 10 ) { x = y ++; z = ++y;}
printf ("x = %d y = %d z = %d\n", x, y, z);
for ( y =1; y < 10; y++ ) x = y;
printf (" x= %d y = %d\n", x, y);
for ( y = 1; ( x = y ) < 10; y++ );
printf ("x = %d y = %d\n", x, y);
for ( x = 0, y = 1000; y > 1; x++, y /= 10 )
printf ("x = %d y = %d\n", x, y);
}
```

3.13. Сравнить семантику операторов case в Паскале и switch в Си.

3.14. Верны ли следующие утверждения:

a) «любое выражение в Си может быть преобразовано в оператор добавлением к нему точки с запятой (;) »

b) «пустой оператор в Си - это отсутствие каких-либо символов в том месте конструкции, где по синтаксису может находиться оператор»

c) «составной оператор (блок) в Си - это совокупность операторов, заключенная в фигурные скобки { }»

d) «оператор присваивания в Си - это выражение вида

переменная = выражение»

e) «тип выражения в условии в операторе if , в условии завершения цикла в операторах цикла может быть скалярным (т.е. любым целочисленным, любым вещественным либо указателем)»

f) «в блоке описания/объявления и операторы могут располагаться в любом порядке; единственное требование - использование не должно предшествовать описанию/объявлению»

e) «цикл for (; ;) является бесконечным циклом, и поэтому его использование в Си запрещено»

f) «семантика операторов цикла while и do-while в Си различается только тем, что тело цикла while может не выполниться ни разу, а тело цикла do-while выполнится хотя бы один раз.»

g) «каждая ветвь в операторе switch должна быть помечена одной или несколькими различными целочисленными константами или константными выражениями»

h) «если в операторе switch нет ветви default, то значение выражения выбора должно совпадать с одной из констант ветвей case»

i) «в операторе switch ветви case и ветвь default можно располагать в любом порядке»

3.15. Верно ли утверждение: « действие оператора continue; в приведенных ниже примерах эквивалентно действию оператора go to next; ».

```
while ( E ) { S; ... continue; ... S; next: ; }
do { S; ... continue; ... S; } while ( E ); next: ; ...
for ( E1; E2; E3) { S; ... continue; ... S; next: ; }
while ( E) { S; ... for (E1;E2;E3) { S; ... continue; ... S; } ... S; next:; }
while ( E) { S; ... for (E1;E2;E3) { S; ... continue; ... S; next: ; } ... S;}
switch ( E ) { case C1: S;
     case C2: S; continue;
     case C3: S; }
next:; ...
switch ( E ) {  case C1: S;
      case C2: S; continue;
      case C3: next: S; }
next: switch ( E ) { case C1: S;
   case C2: S; continue;
   case C3: S; }
```

Замечание: здесь E, E1, E2, E3 - выражения допустимого в этом случае типа ; S - произвольный оператор; C1, C2 C3 - константы подходящего типа.

3.16. Верно ли утверждение: « действие оператора break; в приведенных ниже примерах эквивалентно действию оператора go to next; ».

a) while (E) { S; ... break; ... S; next: ; }
b) while (E) { S; ... break; ... S; } next: ; ...
c) do { S; ... break; ... S; } while (E); next: ; ...
d) for (E1; E2; E3) { S; ... break; ... S; next: ; }
e) while (E) { S; ... for (E1; E2; E3) { S; ... break; ... S; } next: ; ... S; }
f) while (E) { S; ... for (E1; E2; E3) { S; ... break; ... S; } ... S; next: ; }
g) while (E) { S; ... for (E1; E2; E3) { S; ... break; ... S; } ... S; } next: ;
h) switch (E) { case C1: S;
  case C2: S; break;
  case C3: S; }
next:; ...
i) switch (E) { case C1: S;
    case C2: S; break; S;
    case C3: next: S; }

Замечание: здесь E, E1, E2, E3 - выражения допустимого в этом случае типа; S - произвольный оператор; C1, C2 C3 - константы подходящего типа.

4. ОБРАБОТКА ЧИСЛОВЫХ ДАННЫХ

Замечание: при решении некоторых задач этого раздела необходимы минимальные знания о «стандартном» вводе и выводе целых и вещественных чисел.

4.1. Для данных чисел a, b и c определить, сколько корней имеет уравнение ax2+bx+c = 0, и распечатать их. Если уравнение имеет комплексные корни, то распечатать их в виде v ± iw.

4.2. Подсчитать количество натуральных чисел n (111 ≤ n ≤ 999), в записи которых есть две одинаковые цифры.

4.3. Подсчитать количество натуральных чисел n (102 ≤ n ≤ 987), в которых все три цифры различны.

4.4. Подсчитать количество натуральных чисел n (11 ≤ n ≤ 999), являющихся палиндромами, и распечатать их.

4.5. Подсчитать количество цифр в десятичной записи целого неотрицательного числа n.

4.6. Определить, верно ли, что куб суммы цифр натурального числа n равен n2.

4.7. Определить, является ли натуральное число n степенью числа 3.

4.8. Для данного вещественного числа a среди чисел 1, 1 + (1/2), 1 + (1/2) + (1/3), ... найти первое, большее a.

4.9. Для данного вещественного положительного числа a найти наименьшее целое положительное n такое, что 1 + 1/2 +1/3+ ... + 1/n > a.

4.10. Даны натуральное число n и вещественное число x. Среди чисел exp(cos(x2k))sin(x3k) (k = 1, 2, ..., n) найти ближайшее к какому-нибудь целому.

4.11. Дано натуральное число n. Найти значение числа, полученного следующим образом: из записи числа n выбросить цифры 0 и 5, оставив прежним порядок остальных цифр.

4.12.

4.13. Дано натуральное число n. Получить все такие натуральные q, что n делится на q2 и не делится на q3.

4.14. Дано натуральное число n. Получить все его натуральные делители.

4.15. Дано целое число m > 1.Получить наибольшее целое k, при котором 4k < m.

4.16. Дано натуральное число n. Получить наименьшее число вида 2r, превосходящее n.

4.17. Распечатать первые n простых чисел (p - простое число, если p >= 2 и делится только на 1 и на себя).

4.18. Даны вещественные числа x и y (x > 0, y > 1). Получить целое число k (положительное, отрицательное или равное нулю), удовлетворяющее условию yk-1 ≤ x < yk.

4.19. Распечатать первые n чисел Фибоначчи (f0 = 1; f1 = 1; fk+1 = fk-1+ fk; k = 1, 2, 3,...)

4.20. Вычислить с точностью eps > 0 значение «золотого сечения» - 0.5∗(1+√5) - предел последовательности { qi }при i → ∞ qi = fi / fi-1, i = 2, 3,...где fi - числа Фибоначчи (см. предыдущую задачу). Считать, что требуемая точность достигнута, если | qi-qi+1| < eps.

4.21. Распечатать числа Фибоначчи (см. задачу 3.34), являющиеся простыми числами со значениями меньше n.

4.22. Вычислить с точностью eps > 0 значение числа e - предел последовательности { xi }при i → ∞ xi = (1+1/i)i, i = 1, 2, ... Считать, что требуемая точность достигнута, если | xi-xi+1| < eps.

4.23. Вычислить значение ∑ i ! для i, изменяющихся от 1 до n. Воспользоваться соотношением ∑ i ! = 1 + 1∗2 + 1∗2∗3 +...+ 1∗2∗3∗...∗n = 1+2∗(1+3∗(1+ +n∗(1)...)).

4.24. Пусть a0 и b0 - положительные вещественные числа. Соотношениями an+1 = √(anbn) ; bn+1 = (an+bn) / 2 при n = 0, 1, 2, ... задаются две бесконечные числовые последовательности {an}и {bn}, которые сходятся к общему пределу M(a0,b0), называемому арифметико-геометрическим средним чисел a0 и b0. Найти приближенное значение M(a0,b0) с точностью eps > 0. Поскольку при a0 < b0 ai < bi и, более того, a0 < a1 < ... < ai < ... bi < ... < b1 < b0 , то в качестве подходящего критерия прекращения вычислений можно использовать соотношение | ai - bi | < eps.

4.25. Вычислить квадратные корни вещественных чисел x = 2.0, 3.0, ... , 100.0. Распечатать значения x, √x, количество итераций, необходимых для вычисления корня с точностью eps > 0.

4.26. Для a > 0 величина √a вычисляется следующим образом:

4.27. a0 = 1; ai+1 = 0.5∗(ai+a/ai) i = 0, 1, 2,...

4.28. Считать, что требуемая точность достигнута, если | ai-ai+1| < eps.

4.29. Найти приближенное значение числа π с точностью eps > 0. Для этого можно использовать представление числа 2/π в виде произведения корней √(1/2) *√(1/2+1/2√(1/2))*√(1/2+ 1/2√(1/2+1/2√(1/2)))*... . Вычисления прекращаются, когда два следующих друг за другом приближения для числа π будут отличаться меньше, чем на eps.

4.30. Для данного вещественного числа x и натурального n вычислить:

a) sin x + sin2x + ... + sinnx

b) sin x + sinx2 + ... + sinxn

c) sin x + sin(sin x) + ... + sin (sin (... sin(sin x) ...))

4.31. Алгоритм Евклида нахождения наибольшего общего делителя (НОД) неотрицательных целых чисел основан на следующих свойствах этой величины: пусть m и n - одновременно не равные нулю целые неотрицательные числа и m ≥ n. Тогда, если n = 0, то НОД(n, m) = m, а если n ≠ 0, то для чисел m, n, и r, где r - остаток от деления m на n, выполняется равенство НОД(m, n) = НОД(n, r). Используя алгоритм Евклида, определить наибольший общий делитель неотрицательных целых чисел a и b.

4.32. Вычислить 1 - 1/2 + 1/3 - 1/4 + ...+1/9999 - 1/10000 следующими способами:

a). последовательно слева направо;

b). последовательно справа налево;

c). последовательно слева направо вычисляются 1 +1/3 + 1/5 + ... + 1/9999 и 1/2 + 1/4 + ... + 1/10000, затем второе значение вычитается из первого;

d). последовательно справа налево вычисляются 1 +1/3 + 1/5 + ... + 1/9999 и 1/2 + 1/4 + ... + 1/10000, затем второе значение вычитается из первого.

4.33. Сравнить и объяснить полученные результаты.

4.34. Натуральное число называется совершенным, если оно равно сумме всех своих делителей, за исключением самого себя. Дано натуральное число n. Получить все совершенные числа, меньшие n.

4.35. Определить, является ли число простых чисел, меньших 10000, простым числом.

4.36. Если p и q - простые числа и q = p+2, то они называются простыми сдвоенными числами или “близнецами” (twin primes). Например, 3 и 5 - такие простые числа. Распечатать все простые сдвоенные числа, меньшие N.

5. ОБРАБОТКА СИМВОЛЬНЫХ ДАННЫХ

Замечание: при решении некоторых задач этого раздела необходимы минимальные знания о «стандартном» вводе и выводе литер.

5.1 Пусть во входном потоке находится последовательность литер, заканчивающаяся точкой (кодировка ASCII):

a) определить, сколько раз в этой последовательности встречается символ ‘a’;

b) определить, сколько символов ‘e’ предшествует первому вхождению символа ‘u’ (либо сколько всего символов ‘e’ в этой последовательности, если она не содержит символа ‘u’);

c) выяснить, есть ли в данной последовательности хотя бы одна пара символов-соседей ‘n’ и ‘o’, т.е. образующих сочетание ‘n’ ‘o’ либо ‘o’ ‘n’;

d) выяснить, чередуются ли в данной последовательности символы ‘+’ и ‘-‘, и сколько раз каждый из этих символов входит в эту последовательность;

e) выяснить, сколько раз в данную последовательность входит группа подряд идущих символов, образующих слово C++;

f) выяснить, есть ли среди символов этой последовательности символы, образующие слово char;

g) выяснить, есть ли в данной последовательности фрагмент из подряд идущих литер, образующий начало латинского алфавита (строчные буквы), и какова его длина. Если таких фрагментов несколько, найти длину наибольшего из них. Если такого фрагмента нет, то считать длину равной нулю;

h) выяснить, есть ли в данной последовательности фрагменты из подряд идущих цифр, изображающие целые числа без знака. Найти значение наибольшего из этих чисел. Если в этой последовательности нет ни одной цифры, то считать, что это значение равно нулю;

i) определить, имеет ли данная последовательность символов структуру, которая может быть описана с помощью следующих правил:

последовательность ::= слагаемое + последовательность | слагаемое

слагаемое ::= идентификатор | целое

идентификатор ::= буква | идентификатор буква | идентификатор цифра

буква ::= A | B | C | D | E | F | G | H | I | J | K

цифра ::= 0 | 1 | 2 | 3 | 4 | 5

целое ::= цифра | целое цифра

5.2 Пусть во входном потоке находится последовательность литер, заканчивающаяся точкой (кодировка ASCII). Вывести в выходной поток последовательность литер, измененную следующим образом:

a) заменить все символы ‘?’ на’!’;

b) удалить все символы ‘-‘ и удвоить все символы ‘&’;

c) удалить все символы, не являющиеся строчными латинскими буквами;

d) заменить все прописные латинские буквы строчными (другие символы копировать в выходной поток без изменения);

e) заменить все строчные латинские буквы прописными (другие символы копировать в выходной поток без изменения);

f) каждую группу рядом стоящих символов ‘+’ заменить одним таким символом;

g) каждую группу из n рядом стоящих символов ‘*’ заменить группой из n/2 рядом стоящих символов ‘+’ (n >= 2); одиночные ‘*’ копировать в выходной поток без изменения;

h) удалить из каждой группы подряд идущих цифр все начальные незначащие нули (если группа состоит только из нулей, то заменить эту группу одним нулем);

i) удалить все комбинации символов the;

j) оставить только те группы цифр, которые составлены из подряд идущих цифр с возрастающими значениями; все остальные цифры и группы цифр удалить (другие символы копировать в выходной поток без изменения);

k) заменить все комбинации символов child комбинациями символов children;

l) удалить группы символов, расположенные между фигурными скобками { и }. Скобки тоже должны быть удалены. Предполагается, что скобки сбалансированы, и внутри каждой пары скобок других фигурных скобок нет.

5.3 Пусть во входном потоке находится последовательность литер, заканчивающаяся маркером конца $ (кодировка ASCII). Вывести в выходной поток последовательность литер, измененную следующим образом:

a) удалить из каждой группы подряд идущих цифр, в которой более двух цифр и которой предшествует точка, все цифры, начиная с третьей (например, a+12.3456-b-0.456789+1.3-45678 преобразуется в a+12.34-b-0.45+1.3-45678);

b) удалить из каждой группы цифр, которой не предшествует точка, все начальные нули (кроме последнего, если за ним идет точка либо в этой группе нет других цифр, кроме нулей ; например, a-000123+bc+0000.0008-0000+0001.07 преобразуется в a-123+bc+0.0008-0+1.07).

6. ФУНКЦИИ И СТРУКТУРА ПРОГРАММЫ

6.1 Перечислите все существенные изменения, внесенные стандартом ANSI в правила объявления и описания функций. Какова цель этих изменений?

6.2 Перечислить все случаи, когда в Си используется тип void. Дать определение этого типа.

6.3 Перечислить классы памяти, определенные в Си. Что определяет класс памяти? В каких случаях и каким образом класс памяти определяется по умолчанию? Привести примеры.

6.4 Определен ли в Си класс памяти для функций? Если определен, то каким образом; если нет, то почему.

6.5 Допустима ли в Си вложенность функций? Можно ли в Си каким-то образом управлять видимостью функций?

6.6 Объяснить, чем различаются описание (объявление, declaration) и определение (definition) – по терминологии Б. Кернигана и Д. Ритчи [1, см. стр.71]. Привести примеры.

6.7 Эквивалентны ли следующие объявления функций:

a) double f ( ); и double f (void);
b) char g (int i, char c); и char g (int, char);
c) h (double x); и int h (double x);
d) void h (int); и h (int);
e) extern int q (int); и int q (int);
f) static void s (char c); и void s (char c);

6.8 Определить, какие конструкции являются определениями, а какие описаниями; где они могут располагаться и что обозначают:

int i;	char c = ‘a’;	extern int f (int, char);
static int j;	register int b;	double g( ) { return 3.141592; };
extern long k;	int h (int i);	static char q(int, double);
auto short n;	s();	static void p(int i) { };

6.9 Верны ли следующие утверждения:

a) «тип выражения в операторе return должен совпадать с типом результата функции»

b) «функция, которая не возвращает результата (тип результата void), может не содержать оператор return;»

c) «функция, которая возвращает результат, может не содержать оператор return E; но вызывает другую функцию, которая содержит такой оператор»

d) «функция, которая возвращает результат, может содержать несколько операторов return E; »

e) «в Си аргументы функции всегда передаются по значению»

f) «в теле одной функции могут находиться два разных оператора, помеченных одинаковыми метками, если эти операторы находятся в разных блоках»; например,

```
void f(void)
  { ... label: S1; ...
       { ... label: S2; ... goto label; ...}
    ... goto label; ...
  }
```

g) «в Си нельзя использовать две (или более) взаимно рекурсивных функций»; например, если есть void f(void){... g();...} и void g(void){...f();...}, то программа, использующая такие функции, будет ошибочной.

h) «в Си можно войти в блок, минуя его заголовок; при этом память под локальные переменные, описанные в этом блоке, будет отведена, но инициализация (если она есть) выполняться не будет»

i) «любая функция, описанная в каком-либо файле, входящем в состав программы, может быть использована в этом и любом другом файле этой программы»

j) «в этом фрагменте программы нет ошибок »

```
#include <stdio.h>
int f(void) { return 100;}
void g(void) { printf("O.K.\n");}
```

```
main()
   { int i, j;
     i = f();
     j = g(), f();
     g(); f();
     printf("i=%d, j=%d f=%d\n", i, j, f());
   }
```

6.10 В каких случаях в Си возможна инициализация переменных? Когда она происходит? Как определяется инициализация по умолчанию?

6.11 Допустимо ли в Си? Если "да" - опишите семантику этих действий; если "нет" - объясните почему.

a)
```
#include <stdio.h>
main()
   { int i; int sum = 0;
     for ( i = 1; i <= 3; i++)
         { sum += i;
             { int i;
               for ( i = 1; i <= 5; i++)
                    sum *= i;
             }
         }
     printf("sum=%d\n", sum);
   }
```

b)
```
#include <stdio.h>
main()
   { int i; int sum = 0;
     for ( i = 1; i <= 3; i++)
         { sum += i;
             { for ( i = 1; i <= 5; i++)
                    sum *= i;
             }
         }
     printf("sum=%d\n", sum);
   }
```

c)
```
#include <stdio.h>
main()
   { double x;
     scanf("%f", &x);
     if ( x >= 0 ) goto ok;
         { double y = 5.0;
           x = x ? x : -x;
           ok: y+=sqrt(x);
           printf("y = %f\n",y);}
         }
   }
```

d)
```
#include <stdio.h>
main()
   { double x;
     scanf("%f", &x);
     if ( x >= 0 ) goto ok;
         { double y;
           x = x ? x : -x;
           ok: y = sqrt(x);
           printf("y = %f\n",y);
         }
   }
```

6.12 Допустимо ли в Си? Если "да" - опишите семантику этих действий; если "нет" - объясните почему.

a) файл f1:
```
#include<stdio.h>
main()
   { extern int f(int);
     int i;
     i = f(g(5));
     printf("i = %d\n", i);
```

b) файл f1:
```
#include<stdio.h>
static int f(void)
   { int k;
     scanf("%d", &k); return k; }
extern int g(int);
main()
```

```
    }
  int g(int k)
    { k++;
      return f(k);
    }
```

файл f2:

```
  int f(int i) { return i*i; }
```

c) файл f1:

```
#include<stdio.h>
  extern int i; extern void f(void);
  main()
    { f();
      printf("i = %d\n", i);
      f();
      printf("i = %d\n", i);
    }
```

файл f2:

```
  int i = 1;
  void f(void) { i++; }
```

```
      { int i;
      i = f(); j = g(i);
      printf("i=%d,j=%d\n", i,j);
     }
```

файл f2:

```
  extern int f(void);
  int g(int i)
        { return  f()+i; }
```

d) файл f1:

```
#include<stdio.h>
  extern int f(void);
  int i;
  main()
     { printf("res = %d\n", i+f()+f());
      }
```

файл f2:

```
    int f(void)
  { static int i = 10;
    return i--;
   }
```

6.13 Что напечатает следующая программа?

a).

```
#include <stdio.h>
   int i = 0; void f(void);
   main()
       { int i =1;
         printf("i1 = %d\n", i);
         f();
           { int i = 2; printf("i4 = %d\n", i);
                { i +=1; printf("i5 = %d\n", i); }
             printf("i6 = %d\n", i);
           }
         printf("i7 = %d\n", i);
       }
   void f(void)
      { printf("i2 = %d\n", i);
        i = i + 10; printf("i3 = %d\n", i); }
```

b).

```
#include <stdio.h>
  int f( int ); int b = 10;
  main()
     { int c = 3;
       b = f(c);
       printf("c=%d b=%d\n",c,b);
     }
            int f( int b )
     { int c = 5;
       c--; b++;
       printf("c=%d b=%d\n",c,b);
           return c+b;
     }
```

c).

```
#include <stdio.h>
   char g ( char c);
   int f ( int i, char c)
     { int k = i+4; char b = g(c) + i;
       printf("c1 = %c k0 = %d\n", c, k);
         { int i = 3; k += i;
```

d).

```
#include <stdio.h>
  int abc( int ); int x;
   main()
        { int b;
          b = abc(x);
    printf("b1=%d x1=%d\n",b,x);
```

```
        printf("i1 = %d k1 = %d\n", i, k);          x = abc(b);
        c = g('b'); printf("c2 = %c\n", c);         printf("b2=%d
x2=%d\n",b,x);
        }                                           }
    i++; printf("i2 = %d k2 = %d\n", i, k);     int abc( int b )
    return i*(b-c); }                           { static int x = 5;
char g(char c)                                    x += 7; b++;
  { c = c + 1; return c; }                        printf("b0=%d x0=%d\n",b,x);
main()                                            return x+b;
  { int k =2; char c = 'a'; k = f (k, c);       }
    printf("c3 = %c k3 = %d\n", c, k);
  }
```

e).
```
#include <stdio.h>
int i = 1;
int reset ( void );
int next ( int );
int last ( int );
int new ( int );
main()
  { int i, j; i = reset();
    for ( j = 1; j <= 3; j++ )
      { printf("i=%d j=%d\n", i, j);
        printf("%d %d\n", next(i), last(i));
        printf("%d\n", new(i+j));
      } }
int reset ( void ) { return i; }
int next ( int j ) { return j = i++; }
int last ( int j )
  { static int i = 10; return j = i--; }
int new ( int i )
  { int j = 10; return i = j += i; }
```

f).
```
#include <stdio.h>
int i = 1;
int reset ( void );
int next ( void );
int last ( void );
int new ( int );
main()
 { int i, j; i = reset();
   for ( j = 1; j <= 3; j++ )
     { printf("i=%d j=%d\n", i, j);
       printf("%d\n", next(),);
       printf("%d\n", last());
       printf("%d\n", new(i+j)); } }
```

в файле f1:
```
static int i =10;
int next ( void ) { return i += 1; }
int last ( void ) { return i -= 1; }
int new ( int i)
  { static int j = 5; return i=j+=i; }
```

в файле f2:
```
extern int i;
int reset ( void ) { return i; }
```

6.14 Программа. Описать рекурсивную функцию вычисления n! - факториала числа n, основанную на соотношении n! = n∗(n-1)!. С ее помощью найти факториалы натуральных чисел от1 до 10.

6.15 Программа. Описать рекурсивную функцию вычисления x^n для вещественного x ($x \neq 0$) и целого n:

$$x^n = \begin{cases} 1 & \text{при } n = 0 \\ 1/x^{|n|} & \text{при } n < 0 \\ x * x^{n-1} & \text{при } n > 0 \end{cases}$$

Протестировать эту функцию на подходящих наборах входных данных.

6.16 Программа. Описать рекурсивную функцию вычисления НОД(n,m) - наибольшего общего делителя неотрицательных целых чисел n и m, основанную на соотношении НОД(n,m) = НОД(m,r), где r - остаток от деления n на m (см. задачу 3.43). С ее помощью найти наибольший общий делитель натуральных чисел a и b. Сравнить эффективность рекурсивной и нерекурсивной функций вычисления НОД.

6.17 Программа. Описать рекурсивную функцию вычисления НОД (n_1, n_2, n_3, ..., n_m), воспользовавшись для этого соотношением: НОД (n_1, n_2, n_3, ... , n_k) = НОД (НОД (n_1, n_2, n_3, ... , n_{k-1}), n_k), k = 3, 4, ... , m. С ее помощью найти НОД (a_1, a_2, a_3, ..., a_{10}).

6.18 Программа. Описать рекурсивную функцию вычисления n–ого числа Фибоначчи: $f_0 = 1$; $f_1 = 1$; $f_{j+1} = f_{j-1} + f_j$; j = 1, 2, 3,….С ее помощью вычислить 100-ое число Фибоначчи.

6.19 Программа. Описать рекурсивную функцию вычисления значения A(n,m) - функции Аккермана для неотрицательных целых чисел n и m:

$$A(n,m) = \begin{cases} m+1 & \text{если } n = 0 \\ A(n-1,1) & \text{если } n \neq 0,\ m = 0 \\ A(n-1, A(n, m-1)) & \text{если } n>0,\ m>0 \end{cases}$$

С помощью этой функции найти значение A(5,8).

7. УКАЗАТЕЛИ И МАССИВЫ

7.1 Допустимо ли в Си? Если "да" - опишите семантику каждого правильного действия (не принимая во внимание ошибочные); если "нет" - объясните почему.

a). . . .

```
int i,* p, j, *q;
p = &i;              q = &p;
j = *p = 1;          q = p-1;      *p += 1;
i = *++q + *p;       q -= 1;              i = *q ++ + *q;
printf("i=%d, j=%d, *p=%d, *q=%d \n", i, j, *p, *q);
```

b) . . .

```
int x = 1, y; char c = 'a';
int *pi, *qi; char *pc;
pi = &x;     *pi = 3;     y = *pi;     *pi = c;     qi = pi;
pc = qi;     *qi+=1;      pi++;        *(- - pi) = 5; y = *qi+1;
pc = &c;     ++*pc;       (*pc)++;     *pc++;       *pc+=1;
x = (int)pi; pi=(int*)pc; pi=(int*)x;  x = 1+ *pi;  pc=(char*)pi;
c = *pc;     pc = &y;     x = qi – pi; qi = 0;              qi+=pi;
y = &pi;     y = (int)&pi;        pi = pi +5;  *(pi+1)=0;  pi=&(x+0);
```

7.2 К любому ли объекту в Си можно применять операцию взятия адреса & ?

7. 3 Допустимо ли в Си? Если "да" - опишите семантику этих действий; если "нет" - объясните почему.

a) int i = 2; const int j = 5;
int *pi;
const int *pci;
int *const cpi;
const int * const cpci;
pi = &i; pci = &j; cpi = &i; cpci = &j; pci = &i;
pi = (int*)&j; i = *pci + *pi; *pci = 3;
*pi = 3; i=*pci++; *(cpi++)=5; *cpi++;

b) int f(const int i, int j) { j++; return i+j; }
main()
{ int a, b; const int c = 5;
scanf("%d", &a);
b = f(c,a); printf("a=%d, b=%d, c=%d \n", a, b, c);
b = f(c,c); printf("a=%d, b=%d, c=%d \n", a, b, c);
b = f(a,a); printf("a=%d, b=%d, c=%d \n", a, b, c);
b = f(a,c); printf("a=%d, b=%d, c=%d \n", a, b, c);

7.4 Пусть целочисленный массив a соддержит 100 элементов. Верно ли решена задача: "написать фрагмент программы, выполняющий суммирование всех элементов массива a".

a) int a[100], sum, i;
sum = 0;
for (i = 0; i < 100; ++i) sum += a[i];

b) int a[100], *p, sum;
sum = 0;
for (p = a; p < &a[100]; ++p) sum = sum + *p;

c) int a[100], *p, sum;
sum = 0;
for (p = &a[0]; p < &a[100]; p++) sum += *p;

d) int a[100], sum, i;
sum = 0;
for (i = 0; i < 100; ++i) sum += *(a+i);

e) int a[100], sum, i;
sum = 0;
for (i = 0; i < 100; ++a, ++i) sum += *a;

f) int a[100], *p, sum, i;
sum = 0;
for (i = 0, p = a; i < 100; ++i) sum += p[i];

g) int a[100], *p, sum, i;
sum = 0;

for (i = 0, p = a; i < 100; ++i) sum += ∗(p+i);

7.5 Допустимо ли в Си? Если "да" - опишите семантику каждого правильного действия (не принимая во внимание ошибочные); если "нет" - объясните почему.

a) . . .

```
int  a[5] = { 1, 2, 3, 4, 5 };
int  ∗p, x, ∗q , i;
p = a + 2;     ∗ (p+2) = 7;
∗a += 3;       q=∗&p-1;
x = ++ p - q ++;     x += ++ ∗p; x=∗p-- + ∗p++;
for (i = 0; i < 5; i++) printf("a [%d]=%d", i, a[ i ] ); printf("\n");
printf("x=%d, ∗p=%d, ∗q=%d \n", x, ∗p, ∗q);
```

b) . . .

```
char ∗str = "abcdef";
char ∗p, ∗q, ∗r; int k;
p = str;         q = 0;          p++;
k = p - str;    r = p+k;
if ( k && p || q ) q = str + 6;
p = q ? r : q;          ∗(p-1) = ‘a’;∗r = ‘x’;
printf("str: %s\n", str);
```

c) . . .

```
char s[ ] = "0123456";
int ∗pi; char ∗pc1, ∗pc2;
pc2 = s;
pc1 = s + ∗(s+strlen(s) - 1) - ‘0’;
pi = ( int∗ ) pc2;     ∗pc1-- = ‘8’;
if ( pc1 - pc2 < 3 ) pc1 = pc2 = pi; else pc1 = ( pc1+pc2 )/2;
if ( s == pc2 ) ∗pc1 = ∗pc2 + 1; else ∗pc1 = ‘9’;
printf("s: %s\n", s);
```

d) . . .

```
int i; char ∗c; int ∗pi;
i = ‘a’;
pi = &i;       c = (char∗)pi + 3;  printf("c1=%c", ∗c);
i <<= 8;       c--;                printf("c2=%c\n", ∗c);
```

e) . . .

```
char c1, c2; short i;
char ∗pc; short ∗ps;
c1 = ‘1’;      c2 = ‘2’;      ps =&i;
pc = (char∗)ps;      ∗pc = c1;      pc++;          ∗pc = c2;
printf("i = %hd\n", i);
```

7.6 Эквивалентны ли следующие фрагменты программы на Си?

```
a[ i ] /= k+m                    и          a[ i ] = a[ i ]/k+m
```

a[i] /= k+m и a[i] = a[i]/(k+m)
a[i++]+=3 и a[i++] = a[i++]+3
a[i++]+=3 и a[i] = a[i++]+3
a[i++]+=3 и a[i++] = a[i]+3
a[i++]+=3 и a[i] = a[i]+3; i++;

7.7 Что напечатает следующая программа?

```
#include <stdio.h>
char str[ ] = "SSSWILTECH1\1\11W\1WALLMP1";
main()
{ int i, c;
  for ( i = 2; ( c = str [ i ] ) != '\0'; i++) {
switch (c) {
  case 'a': putchar('i'); continue;
  case '1': break;
  case 1: while ( ( c = str [++ i ] ) != '\1' && c != '\0');
  case 9: putchar('S');
  case 'E': case 'L': continue;
  default: putchar(c); continue; }
putchar(' '); }
  putchar('\n');
}
```

7.8 Что напечатает следующая программа?

```
#include <stdio.h>
int a[ ] = { 0, 1, 2, 3, 4 };
main()
   { int i, *p;
     for ( i = 0; i <= 4; i++ ) printf("a[ i ]=%d  ", a[ i ]);  printf("\n");
     for ( p = &a[0]; p <= &a[4]; p++ ) printf("*p=%d  ", *p);  printf("\n");
    for ( p = &a[0], i = 0 ; i <= 4; i++ ) printf("p[ i ]=%d  ", p[ i ]);  printf("\n");
     for ( p = a, i = 0; p+i <= a+4; i++ ) printf("* (p+i)=%d ", * (p+i));
     printf("\n");
     for ( p = a+4; p >= a; p-- ) printf("*p=%d  ", *p );  printf("\n");
     for ( p = a+4, i=0; i <= 4; i++ ) printf("p[ -i ]=%d  ", p[ -i ]);
     printf("\n");
     for ( p = a+4; p >= a; p -- ) printf("a[ p - a ]=%d  ", a[ p - a ]);
     printf("\n");
   }
```

7.9 Что напечатает следующая программа?

```
      #include <stdio.h>
      int a[ ] = { 8, 7, 6, 5, 4 };
      int  *p[ ] = { a, a+1, a+2, a+3, a+4 };
      int  **pp = p;
      main()
```

```
    { printf("*a=%d  **p=%d  **pp=%d\n", *a, **p, **pp );
      pp++;
      printf("pp-p=%d  *pp-a=%d  **pp=%d\n", pp-p, *pp-a, **pp );
      ++*pp;
      printf("pp-p=%d  *pp-a=%d  **pp=%d\n", pp-p, *pp-a, **pp );
      pp = p;
      ++**pp;
      printf("pp-p=%d  *pp-a=%d  **pp=%d\n", pp-p, *pp-a, **pp );
    }
```

7.10 Что напечатает следующая программа?

```
#include <stdio.h>
int a[ 3 ][ 3 ] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
int  *pa[ 3 ] = { a[ 0 ], a[ 1 ], a[ 2 ] };
int  *p = a[ 0 ];
main()
        { int i;
          for ( i = 0; i < 3; i ++ )
printf(" a[ i ][ 2 – i ]=%d  *a[ i ]=%d  *(*(a+i)+i)=%d\n",
a[ i ][ 2 – i ], *a[ i ], *(*(a+i)+i));
          for ( i = 0; i < 3; i ++ )
printf("*pa[ i ]=%d  p[ i ]=%d \n", *pa[ i ], p[ i ] );
}
```

7.11 Что напечатает следующая программа?

```
#include <stdio.h>
char *c[ ] = { "ENTER", "NEW", "POINT", "FIRST" };
char ** cp[ ] = { c+3, c+2, c+1, c };
char ***cpp=cp;
main()
          { printf("%s", **++cpp );
            printf("%s  ", * -- *++cpp+3 );
            printf("%s", *cpp[ -2 ]+3 );
            printf("%s\n", cpp[ -1 ][ -1 ]+1 );
```

7.12 Какие соглашения о конце строки существуют в Си и Паскале? Укажите все «за» и «против» явного указания концов строк с помощью null-литеры ‘\0’.

7.13 В чем заключается проблема «висящей» ссылки? Приведите примеры.

7.14 Нужна ли в Си «сборка мусора»? Почему возникает такая проблема и как она решается в Си?

7.15 Прочитайте следующие описания и определения:

```
 int *ip, f( ), *fip( ), (*pfi)( );       char *str[10];     char * (*cp)[5];
```

```
int (*r) ( );                    double (*k)(double,int*);
float * (* (*x) [6] )( );        double (* (* ( y( ) )[ ] )( );
int * (*const *name[9])(void);   char * const p;
```

7.16 Определите переменную x как массив указателей на функцию, имеющую два параметра типа int и возвращающую результат типа указатель на double.

7.17 Определите переменную y как указатель на массив указателей на функцию без параметров, возвращающую результат типа указатель на функцию с одним параметром типа int и результатом типа float.

7.18 Что будет напечатано? Объяснить, почему результат будет таким.

a)
```
#include <stdio.h>
int  try_to_change_it(int);
main()
   { int i = 4, j;
     j = try_to_change_it(i);
     printf("i=%d, j=%d\n", i, j);
   }
int  try_to_change_it(int k)
   { printf("k1=%d\n", k);
     k+=33;
     printf("k2=%d\n", k);
     return k;
   }
```

b)
```
#include <stdio.h>
void compare (int , int *);
main()
  { int i = 4, j = 5;
    compare(i, &j);
      printf("i=%d, j=%d\n", i, j);
    }
void compare (int k, int *m)
  { printf("k1=%d,*m1=%d\n",k, *m);
      k++; (*m)++;
    printf("k2=%d,*m2=%d\n", k, *m);
    }
```

7.19 Верно ли решена задача: « Описать функцию, меняющую местами значения двух переменных символьного типа. Использовать эту функцию для изменения значений символьных переменных a и b.»

a)
```
void swap ( char x, char y)
    { char t; t = x; x = y; y = t;}
main()
    { char a,b;
      scanf("%c%c", &a, &b);
      swap(a,b);
      printf("a=%c,b=%c\n",a,b);
    }
```

b)
```
void swap ( char *x, char *y)
    { char *t; t = x; x = y; y = t;}
main()
    { char a,b;
      scanf("%c%c", &a, &b);
      swap(&a, &b);
        printf("a=%c,b=%c\n",a,b);
    }
```

c)
```
void swap ( char *x, char *y)
    { char t; t = *x; *x = *y; *y = t;}
main()
    { char a,b;
      scanf("%c%c", &a, &b);
      swap(a,b);
      printf("a=%c,b=%c\n",a,b);
    }
```

d)
```
void swap ( char *x, char *y)
    { char t; t = *x; *x = *y; *y = t;}
main()
    { char a,b;
      scanf("%c%c", &a, &b);
      swap(&a, &b);
        printf("a=%c,b=%c\n",a,b);
    }
```

e)
```
void swap ( char x, char y)
    { char *t; t = &x; &x = &y; &y = t;}
main()
    { char a,b;
      scanf("%c%c", &a, &b);
      swap(&a, &b);
      printf("a=%c,b=%c\n",a,b);
    }
```

f)
```
void swap ( char &x, char &y)
    { char t; t = x; x = y; y = t;}
main()
    { char a,b;
      scanf("%c%c", &a, &b);
      swap(a, b);
      printf("a=%c,b=%c\n",a,b);
    }
```

7.20 Допустимо ли в Си? Если "да" - опишите семантику этих действий; если "нет" - объясните почему.

```
int ques ( char *s1, char *s2)
   { while (*s1 && *s2 && *s1++ == *s2++ );
     return *--s1 - *--s2;
    }
```

7.21 Допустимо ли в Си? Если "да" - опишите семантику этих действий; если "нет" - объясните почему.

```
void ques ( char *s1, char *s2, int n)
     { while (*s1 && *s2 && n-- && (*s1 ++ = *s2 ++ ) ); }
```

7.22 Описать функцию, определяющую упорядочены ли строго по возрастанию элементы целочисленного массива из n элементов.

7.23 Описать функцию, определяющую индекс первого элемента целочисленного массива из n элементов, значение которого равно заданному числу x. Если такого элемента в массиве нет, то считать номер равным –1.

7.24 Описать функцию, вычисляющую значение $x_0 + x_0*x_1 + x_0*x_1*x_2 + \ldots + x_0*x_1*x_2*\ldots*x_m$, где x_i - элементы вещественного массива x из n элементов, m - индекс первого отрицательного элемента этого массива либо число n-1, если такого элемента в массиве нет.

7.25 Описать функцию, вычисляющую значение max($x_0 + x_{n-1}$, $x_1 + x_{n-2}$, $x_2 + x_{n-3}$,…, $x_{(n-1)/2} + x_{n/2}$), где x_i - элементы вещественного массива x из n элементов.

7.26 Описать функцию, вычисляющую значение min($x_0 * x_1$, $x_1 * x_2$,$x_2 * x_3$,…, $x_{n-3} * x_{n-2}$, $x_{n-2} * x_{n-1}$), где x_i - элементы вещественного массива x из n элементов.

7.27 Описать функцию, вычисляющую значение $x_0*y_0+x_1*y_1+ \ldots + x_k*y_k$, где x_i – отрицательные элементы вещественного массива a из n элементов, взятые в порядке их следования; y_i – положительные элементы этого массива, взятые в обратном порядке; k = min(p,q), где p – количество положительных элементов массива a, q – количество отрицательных элементов этого массива.

7.28 Описать функцию, которая упорядочивает элементы целочисленного массива, используя следующий алгоритм сортировки:

a) сортировка выбором: находится максимальный элемент массива и переносится в его конец; затем этот метод применяется ко всем элементам массива, кроме последнего (т.к. он уже находится на своем месте.

b) сортировка обменом (метод пузырька): последовательно сравни-ваются пары соседних элементов x_k и x_{k+1} (k = 0, 1, … ,n-2) и, если $x_k > x_{k+1}$, то они переставляются; в результате наибольший элемент окажется на своем месте в конце массива; затем этот метод применяется ко всем элементам, кроме последнего, и т.д.

c) сортировка вставками: пусть первые k элементов массива (от 0 до k-1) уже упорядочены по неубыванию; тогда берется x_k и размещается среди первых k элементов так, чтобы упорядоченными оказались уже k+1 первых элементов; этот метод повторяется при k от 1 до n-1.

7.29 Описать функцию, определяющую индекс первого элемента целочисленного массива из n элементов, значение которого равно заданному числу x. Если такого элемента в массиве нет, то считать номер равным –1. Элементы массива упорядочены по возрастанию; использовать метод двоичного (бинарного) поиска.

7.30 Программа. Описать функцию f(a, n, p), определяющую, чередуются ли положительные и отрицательные элементы в целочисленном массиве a из n элементов и вычисляющую целочисленное значение p. Если элементы чередуются, то p - это сумма положительных элементов, иначе p - это произведение отрицательных элементов. С помощью этой функции провести анализ целочисленного массива x [50].

7.31 Программа. Описать функцию f(a, n, p), определяющую, упорядочены ли строго по возрастанию элементы в целочисленном массиве a из n элементов, и вычисляющую целочисленное значение p. Если элементы упорядочены, то p - это произведение разностей рядом стоящих элементов, иначе p - это количество нарушений порядка в массиве a. С помощью этой функции провести анализ целочисленного массива b [60].

7.32 Программа. Описать функцию f (s, n, x), определяющую, какой символ чаще других встречается в строке s и сколько раз он в нее входит. Если таких символов несколько, то взять первый из них по алфавиту. С помощью этой функции провести анализ строки str.

7.33 Программа. Описать функцию f(s, n, x), определяющую, какой символ реже других (но не нуль раз) встречается в строке s и сколько раз он в нее входит. Если таких символов несколько, то взять первый из них по алфавиту. С помощью этой функции провести анализ строки str.

7.34 Программа. Для целочисленного массива a, содержащего n элементов, описать функцию f(a, n, last, k, nlast), определяющую last - значение последнего из элементов массива a, значение которого принадлежит диапазону

[-k, k], и nlast - индекс этого элемента. С помощью этой функции вычислить соответствующие значения last и nlast для целочисленных массивов x[20] и y[30].

7.35 Программа. Для вещественного массива a, содержащего n элементов, описать функцию G, определяющую значения максимального и минимального элементов этого массива. С помощью этой функции для вещественных массивов x[25] и y[40] вычислить соответствующие значения.

7.36 Описать функцию, которая изменяет заданную строку следующим образом: сначала записывает все элементы с четными индексами, а затем все элементы с нечетными индексами (с сохранением их относительного порядка в каждой группе).

7.37 Описать функцию, которая в заданной строке меняет местами ее первую и вторую половины.

Например, abcdefgh => efghabcd, vwxyz => yzxvw.

7.38 Описать функцию, осуществляющую циклический сдвиг на n позиций вправо элементов целочисленного массива, содержащего m элементов (n<m).

7.39 Описать функцию, осуществляющую циклический сдвиг на n позиций влево элементов целочисленного массива, содержащего m элементов (n<m).

7.40 Написать программу, обнуляющую каждую четную двоичную единицу в коде, размещенном в переменной типа int. Вывести исходные данные и полученный результат в виде, удобном для анализа проведенных преобразований.

7.41 Написать программу, обнуляющую каждую нечетную двоичную единицу в коде, размещенном в переменной типа int. Вывести исходные данные и полученный результат в виде, удобном для анализа проведенных преобразований.

7.42 Описать функцию, которая в каждом элементе беззнакового целочисленного массива заменяет старший байт нулевым кодом, если в этом байте размещен код латинской буквы.

СПИСОК ОСНОВНЫХ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

1. Федеральный государственный образовательный стандарт среднего профессионального образования по специальности 23.02.03 Техническое обслуживание и ремонт автомобильного транспорта (Приказ Минобрнауки России от 22.04.2014г. № 383 "Об утверждении федерального государственного образовательного стандарта среднего профессионального образования по специальности 23.02.03 Техническое обслуживание и ремонт автомобильного транспорта").

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Основная литература

1. Б. Керниган, Д. Ритчи. Язык программирования Си. М., «Финансы и статистика», 1992 г.

2. American National Standard for Information Systems - Programming Language C, X3.159-1989 г.

3. Б. Керниган, Д. Ритчи, А. Фьюэр Язык программирования Си. Задачи по языку Си. М., «Финансы и статистика», 1985 г.

4. Н. Джехани. Программирование на языке Си. М., «Радио и связь», 1988 г.

5. Б. Керниган, Р. Пайк. Универсальная среда программирования UNIX. М., «Финансы и статистика», 1992 г.

Дополнительная литература

1. Колдаев, В. Д., Лупин, С. А. Архитектура ЭВМ: учеб. пособие для сред. проф. Образования. М.: ФОРУМ-ИНФРА-М, 2013 г.

2. Гагарина, Л. Г., Кокорева, Е. В. Введение в теорию алгоритмических языков и компиляторов: учеб. пособие для вузов. М.: Форум, 2013 г.

Перечень ресурсов информационно-телекоммуникационной сети "Интернет"

1. Солдатенко, И.С. Практическое введение в язык программирования Си [Электронный ресурс] : учебное пособие / И.С. Солдатенко, И.В. Попов. — Электрон. дан. — Санкт-Петербург : Лань, 2018. — 132 с. — Режим доступа: https://e.lanbook.com/book/109619. — Загл. с экрана.

2. Пушкарев, А.Н. Языки программирования [Электронный ресурс] : учебно-методическое пособие / А.Н. Пушкарев. — Электрон. дан. — Тюмень : , 2018. — 48 с. — Режим доступа: https://e.lanbook.com/book/110182. — Загл. с экрана.