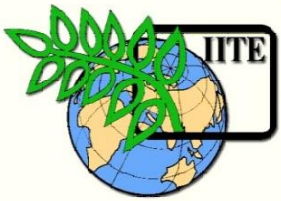

ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ЦИФРОВЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ

Кафедра «Информационные технологии»

СБОРНИК УПРАЖНЕНИЙ

по дисциплине

«Высокоуровневые методы информатики и программирования»

Авторы
Рашидова Е.В.,
Борисова Е.В.

Ростов-на-Дону, 2023

Аннотация

Методические указания предназначены для проведения лабораторных работ по дисциплине «Высоко-уровневые методы информатики и программирования» (для студентов 2 – 3 курса специальностей дневногои заочного отделений).

Каждая лабораторная работа включает набор заданий, методические указания к ним и контрольные вопросы по изучаемой теме. В конце методических указаний приведен перечень рекомендуемой литературы.

Методические указания могут быть использованы для самостоятельной работы.

Авторы

к.ф.-м.н. доц. Рашидова Е.В.
доц., к.т.н. Борисова Е.В.

Оглавление

ЛАБОРАТОРНАЯ РАБОТА №1
«ТИПЫ, ОПЕРАЦИИ, ВЫРАЖЕНИЯ»

Задание:

1.1. 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’
00123	0xffffffL	01A	-‘x’	″x″
‘a’U	0731UL	‘\n’	+0xaf	0X0

1.2. 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+2	0e0	0x1A1.5	05.5	0
0X1E6	0F	1234.56789L	1.0E-10D	3.1415U
1e-2f	-12.3E-6	+10e6	123456L	E-6

1.3. 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
```

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

```
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;
```

1.5. Верно ли решена задача: «значение целочисленной переменной с уменьшить на 1; целочисленной переменной a присвоить значение, равное частному от деления переменной с 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;

1.6. 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 + - * / % >><<& ^ |).

1.7. 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 ) ++
```

1.8. 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
```

1.9. В логике справедливы утверждения:

not (not x) = x

x and true = x

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

Ответ обосновать.

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

1.11. 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)

1.12. Пусть char c; short s; inti; unsigned u; signed char sc;
float f; double d; long lng; unsigned short us; long double ld;
Определитьтипвыражений:
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)

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

a). . . .
inti;
z;
i = (1 || 2) % (1 | 2);
printf (″i = %d\n″, i);
0;
= z + (m != n);
%d %d %d %d\n″,
m, n, z);

b). . . .
int a, b, m, n,
m = n = 5;
z = a = b =
z--, (a = b)
printf (″%d
a, b,

c). . . .
inti = 1;
1.9; int a;
i = i<<i | i;
3.7;
printf (″i = %d\n″, i);
&& 2 || 3) != (int)x;
%d %f\n″, x, a, b);

d). . . .
double x =
double b =
a = b += (1
printf (″%f

e). . . .
int x;
5; y = 10; i = 15;
x = 5; ++ x =10;
= 1);
printf ("%d\n", x);
%d %d\n", i, x ,y);

f). . . .
inti, x, y; x =
x = (y = 0, i
printf("%d

```
, i=1;
%d %d\n", i, x, y);
```

g). . . .

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

i). . . .

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

k). . . .

```
int x, y, z;  x = y = z = 1;
x = y = z = 1;
x += y += z;
++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--);
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);
```

```
( x = y == 0)
printf("%d
```

h). . . .

```
int x = 2, y, z;
x *= 3+2; x
printf ("%d
x = y == z; x
printf ("%d
```

j). . . .

```
int x = 03, y
printf("%d\n",
printf("%d\n",
printf("%d\n",
printf("%d\n",
```

l). . . .

```
int x, y, z, i;
i = ++x ||
i = x++ <= --
```

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

```
. . .
double d; float f; long lng; inti; short s;
```

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

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

```
        double d = 3.2, x; inti = 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);
```

1.16. 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.

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

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

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

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

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

1.22. 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

1.23. 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.

1.24. 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?

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

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

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

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

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

1.30. 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 longdouble;

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 от-

ношения;

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.

ЛАБОРАТОРНАЯ РАБОТА №2
«УСЛОВНЫЙ ОПЕРАТОР И ОПЕРАТОР SWITCH»

Задание:

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

2. 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. Определить какие ошибки встречаются в приведенных ниже фрагментах программы, и какие из фрагментов без ошибок эквивалентны между собой:

a)
```
if x<5 y>=-10
  printf("%d",x+y);
```

b)
```
x=45;y=-10;
if (x<5)
if (y>=-10)
  printf("%d",x+y);
```

c)
```
x=45;y=-10;
if (x<5,y>=-10)
  printf("%d",x+y);
```

d)
```
x=45;y=-10;
if (x<5&&y>=-10)
  printf("%d",x+y);
```

4. Какой из фрагментов программы соответствует решению следующей задачи: если X положительное(не равное нулю) четное целое число, то присвоить X его квадрат, если не четное, то остаток от деления на три.

a)
```
if !(x<=0)
if (x%2==0)
  x=x*x;
else x%=3;
```

b)
```
if (x>0)
if (x%2!=0)
  x=x*x;
else x%=3;
```

c)
```
if !(x%2=0)
if (x>0)
x=x*x;
else x%=3;
```

d)
```
if !(x%2=0)
{if (x>0)
  x=x*x;}
elsex%=3;
```

5. Какие ошибки содержаться в приведенных ниже фрагментах программ:

a)
```
int x;
….
switch (x)
  {
    case 5: x++; break;
    case x>0: x--; break;
    default: x+=66;
```

```
}

b)
int x;
….
switch (x)
  {
    case 5: x+=7; break;
    case 6: case 11: --x; break;
}

c)
int x;
….
switch (x)
  {
    case 5: x*=16;
    case 6: case 11: x-=23; break;
}

d)
int x;
….
switch (5)
  {
    case 5: x+=7; break;
  case 6: case 11: --x; break;
}
```

6. Какие из задач можно решить с помощью оператора switch:

- Если остаток от деления X на 5 равен 2, то присвоить X значение 0, если остаток равен 3, то умножить X на –1.
- Если X равно 5, то присвоить Y квадрат X, если 7, то У присвоить остаток от деления X на 2, в остальных случаях значение Y увеличить на 1.
- Если X больше 5, то увеличить его значение на 1, иначе вывести на печать У. X и Y целые.

7. Определить, имеется ли среди заданных целых чисел A, B, C хотя бы одно чётное.

8. Даны три числа. Вывести на экран те из них, которые принадлежат заданному отрезку [e, f].

9. Определить, есть ли среди цифр заданного целого трёхзначного числа одинаковые.

10. Выбрать наибольшее из трёх заданных чисел.

11. Определить номер квадранта, в котором находится точка с заданными координатами (x, y).

12. Написать программу, которая запрашивает у пользователя номер дня недели и выводит одно из сообщений: «Увы – рабочий день!», «Ура! Суббота!», «Ура! Воскресенье!».

13. Написать программу, которая после ввода с клавиатуры числа (в диапазоне от 1 до 999), обозначает денежную единицу, дописывая слово «рубль» в правильной форме. Например, 12 рублей, 21 рубль и т. д.

14. Написать программу, вычисляющую стоимость междугороднего разговора в соответствии с таблицей:

Город	Код	Цена руб/мин
Владивосток	432	9,20
Москва	095	4,10
Краснодар	861	2,05
Волгоград	844	2,50

Входными данными должны являться код города и число минут.

На выходе мы должны иметь сообщение о стоимости минуты и сумме за разговор.

15. Проверить, поместится ли на диске компьютера музыкальная композиция, которая длится **m** минут и **n** секунд, если свободное дисковое пространство 6 мегабайт, а для записи одной секунды звука необходимо 16 килобайт.

16. Для нормального разведения золотых рыбок необходимо, чтобы на каждую рыбку в аквариуме приходилось не менее 3-х литров воды. По известным объему аквариума и количеству рыбок, в нем содержащихся, определить, является ли аквариум "перенаселенным" или нет, и указать количество рыбок, которых в случае перенаселенности необходимо поместить в другой аквариум.

ЛАБОРАТОРНАЯ РАБОТА №3 «СИНТАКСИС И СЕМАНТИКА ЦИКЛИЧЕСКИХ КОНСТРУКЦИЙ ЯЗЫКА СИ»

Задания:

1. Описать в виде блок-схемы семантику каждого оператора цикла в Си.

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

3. 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);

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

5. Эквивалентны ли следующие фрагменты программы:

a) for (; e2 ;) S; и while (e2) S;

b) for (; ;) S; и while (1) S;

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

6. 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.

7. 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);
}
```

8. Верно ли утверждение: « действие оператора continue; в приведенных ниже примерах эквивалентно действию оператора gotonext; ».

a) while (E) { S; ... continue; ... S; next: ; }
b) do { S; ... continue; ... S; } while (E); next: ; ...
c) for (E1; E2; E3) { S; ... continue; ... S; next: ; }
d) while (E) { S; ... for (E1;E2;E3) { S; ... continue; ... S; } ... S; next:; }
e) while (E) { S; ... for (E1;E2;E3) { S; ... continue; ... S; next: ; } ... S;}

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

9. Верно ли утверждение: « действие оператора break; в приведенных ниже примерах эквивалентно действию оператора gotonext; ».

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: ; }

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

ЗАДАЧИ

1. Подсчитать количество натуральных чисел n ($111 \le n \le 999$), в записи которых есть

a) две одинаковые цифры;

b) только две одинаковые цифры.

2. Подсчитать количество натуральных чисел n ($102 \le n \le 987$), в которых все три цифры различны.

3. Подсчитать количество натуральных чисел n ($11 \le n \le 999$), являющихся палиндромами, и распечатать их.

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

5. Определить, верно ли, что куб суммы цифр натурального числа n равен n^2.

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

7. Для данного вещественного числа ***a*** найти наименьшее натуральное число ***n***, такое что сумма **1 + (1/2) + (1/3)+ . . . + (1/n)** будет большее числа ***a***.

8. Для данного вещественного положительного числа a найти наименьшее натуральное положительное ***n*** такое, что **1 + 1/2 +1/3+ … + 1/n > a**.

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

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

11. Дано натуральное число n. Получить все такие натуральные q, что n делится на q^2 и не делится на q^3.

12. Дано целое число m > 1.Получить наибольшее целое k, при котором $4^k < m$.

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

14. Распечатать первые n (n - задано) чисел Фибоначчи:

$f_0 = 1$; $f_1 = 1$; $f_{k+1} = f_{k-1} + f_k$; k = 1, 2, 3,...

15. Вычислить значение $\sum i!$ для i, изменяющихся от 1 до n. Воспользоваться соотношением $\sum i! = 1 + 1*2 + 1*2*3 +...+ 1*2*3*...*n = 1+2*(1+3*(1+ +n*(1)...))$.

16. Вычислить квадратные корни вещественных чисел x = 2.0, 3.0, ... , 100.0. Распечатать значения x, $\sqrt{x}$. Количество итераций, необходимых для вычисления корня, определяется точностью eps> 0 (eps – задано).

Для a> 0 величина $\sqrt{a}$ вычисляется следующим образом:

$a_0 = 1$; $a_{i+1} = 0.5*(a_i + a/a_i)$ i = 0, 1, 2,...

Считать, что требуемая точность достигнута, если $| a_i - a_{i+1}| <$eps.

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

a) $\sin x + \sin^2 x + ... + \sin^n x$

b) $\sin x + \sin x^2 + ... + \sin x^n$

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

18. 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.

Сравнить и объяснить полученные результаты.

19. 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.

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

ЛАБОРАТОРНАЯ РАБОТА №4 «ФУНКЦИИ. ВВОД И ВЫВОД И ПРЕОБРАЗОВАНИЕСИМВОЛОВ»

I.

Функция getchar() считывает один символ из стандартного входного потока (обычно, ввод с клавиатуры). Чтение происходит после нажатия клавиши ENTER, что позволяет исправлять вводимую информацию (стирать ее, начиная с последних символов клавишей BackSpase). Функция getchar() осуществляет буферизованный ввод символа с отображением на экране (эхопечатью), т.е. можно ввести до нажатия ENTER несколько символов и они будут считываться при следующих обращениях к getchar() (т.к. сохранятся в буфере ввода).

При успешном завершении функция getchar() возвращает символ, преобразованный в целочисленный тип int (т.е. однобайтовый char преобразуется в двухбайтовое целое int). Преобразование позволяет сообщать об ошибке ввода, используя константу EOF (EndOfFile), описанную в файле stdio.h и имеющей целое значение -1 (в шестнадцатеричном представлении FFFF). При вводе с клавиатуры EOF возвращается при нажатии Ctrl+Z.

Если стандартное устройство ввода переназначено на файл, то EOF возвращается при достижении конца файла.

Прототип функции

intgetchar(void);

описан в заголовочном файле стандартного ввода/вывода **<stdio.h>** :

Функция putchar() выводит символ в стандартный выходной поток (обычно, на экран монитора). При успешном завершении putchar() возвращает выведенный символ, а при ошибке - возвращает EOF.

Прототипфункции

intputchar(intch);

описан в заголовочном файле стандартного ввода/вывода **<stdio.h>** :

Пример ввода и вывода символа:

```
#include <stdio.h>
void main()
{
```

```
inta;
a=getchar();
putchar('\n');
putchar(a);
}
```

Задание 1.

Написать программу, которая вводит с клавиатуры несколько (по-своему усмотрению) символов, являющихся строчными латинскими буквами. Предусмотреть "защиту от дурака", т.е. при вводе должна осуществляться проверка, действительно ли введенные символы являются строчными латинскими буквами. Затем преобразовать эти символы в прописные латинские буквы, преобразование оформить в виде функции с прототипом intregist(int); Вывести результат в стандартный поток выхода.

II.

Функция getch() получает символ непосредственно с клавиатуры без эхо-печати и буферизации. Она не требует подтверждения конца ввода с помощью ENTER. Возвращает код, соответствующий нажатой клавише. При нажатие клавиши ENTER возвращает код 13.

На нажатие Ctrl+Z возвращает код 26 (на Ctrl+A – код 1, на Ctrl+B – код 2 и т.д.).

Функцией getch() удобно пользоваться для задержки экрана вывода (до нажатия любой другой клавиши).

Прототип функции **intgetch(void);**

описан в заголовочном файле консольного ввода/вывода **<conio.h>** (нестандартная библиотека, особенность компиляторов Borland)

Функция getche() (буква 'e' от echo - эхо) описана в файле <conio.h>с прототипом

intgetche(void);

Отличие от getch() в отражении на экране вводимого символа (эхо-печать). Управляющие символы таблицы ASCII выводятся на экран функцией getche() в виде специальных обозначений.

ОЧИСТКА ЭКРАНА

Для очистки экрана используется функция clrscr(), описанная в файле **<conio.h>**:

voidclrscr();

После очистки экрана курсор устанавливается в левый верхний угол экрана.

Задание 2.

Написать программу, которая вводит с клавиатуры два символа, являющихся цифрами из интервала [0; 5]. Предусмотреть "защиту от дурака", т.е. при вводе должна осуществляться проверка, действительно ли введенные символы цифры, принадлежащие отрезку [0; 5]. Затем преобразовать эти символы в цифры, на 3 больше, чем исходные, для этого использовать написанную вами функцию с прототипом intadd3(int);

Вывести результат в стандартный поток выхода. Для ввода символов попробовать использовать (при разных запусках) функции getch(), getche(), getchar(). Выяснить различия работы указанных функций.

Задание 3.

Ввести во входной поток последовательность символов, заканчивающуюся 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 символов, 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;

Каждое задание оформить в виде отдельной функции без параметров, возвращающей соответствующее заданию значение, т.ес прототипом

<тип возвращаемого значения> имя функции(void);

ввод описанной выше последовательности последовательность символов должен производиться в теле каждой создаваемой вами функции, предусмотреть вывод на экран сообщения типа "ВВЕДИТЕ ПОСЛЕДОВАТЕЛЬНОСТЬ СИМВОЛОВ". Таким образом, программа к заданию 3 по окончанию его выполнения должна содержать кроме main() 7 созданных вами функций.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Пусть во входном потоке находится последовательность символов, заканчивающаяся точкой (кодировка ASCII). Вывести в выходной поток последовательность символов, 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.

2. Пусть во входном потоке находится последовательность символов, заканчивающаяся маркером конца $ (кодировка ASCII). Вывести в выходной поток последовательность символов, 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).

ЛАБОРАТОРНАЯ РАБОТА №5
«ВВОД - ВЫВОД СТРОК»

Функции scanf и printf являются универсальными функциями форматного ввода и вывода, поэтому их применение приводит к включению в программу довольно значительного по объему кода из библиотеки ввода-вывода. Для ввода-вывода символов и строк проще и экономнее пользоваться специализированными функциями.

Функция ввода символьных строк gets

Функция gets (сокращение от getstring) считывает строку из стандартного входного потока. Ее прототип описан в заголовочном файле stdio.h:

char *gets(char* s),

где **s** - указатель на начало области памяти, в которую будет записываться вводимая строка.

Выделение памяти для хранения строки можно производить, определив массив символов необходимой длины (включая позицию для нуль-терминатора '\0').

Обращение к функции: **gets(s).**

Функция gets читает одну строку символов из стандартного потока ввода stdin, обычно связанного с клавиатурой. Признаком конца ввода для функции gets является нажатие клавиши ENTER (или ввод символа '\n' - «переход на новую строку»). Все набранные до этого момента символы (включая пробелы и символы табуляции) считываются в строку с начальным адресом **s**. Символ '\n' функция gets заменяет на нуль-терминатор '\0' и записывает его в конец строки.

При успешном завершении функция gets возвращает указатель на введенную строку, т.е. **s**. При достижении конца файла (EOF), что при вводе с клавиатуру соответствует нажатию Ctrl+Z, или при обнаружении ошибки gets возвращает NULL-указатель.

Функция вывода символьных строк puts

Функция puts (сокращение от putstring) для вывода символьных строк более проста и менее объемна, чем printf. Ее прототип описан в заголовочном файле stdio.h:

intputs(char* s),

где **s** - указатель на начало области памяти, в которой хранится выводимая строка.

Обращение к функции: **puts(s).**

Функция puts копирует в стандартный поток stdout (обычно связанный с экраном) последовательность символов, начиная с адреса **s** и до первого символа '\0'. Нуль-терминатор при выводе заменяется на символ '\n', что вызывает автоматический переход на новую строку экрана при завершении работы puts.

При успешном завершении функция puts возвращает последний выведенный символ строки. В противном случае - возвращается EOF.

Задание 1.

Составить программу, которая запрашивает ввод строки символов и затем печатает ее.

```
#include <stdio.h>
char string[100];
void main() {
clrscr();
puts("Введите строку текста:\n");
gets(string);
puts("\nБылавведенастрока:"); puts(string);
getch();
}
```

Задание 2.

1. Описать символьный массив str (например, из 50 элементов).
2. С помомщьюputs вывести на экран приглашение "Введите строку символов: ".
3. С помощью getsввести в массив s произвольную строку символов.
4. Вывести на экран текст "Введена строка:" и с новой строки вывести строку str.
5. Вывести на экран текст

Работа выполнена:

студентами группы "шифр группы"

1) Фамилия \ Имя

2) Фамилия \ Имя

Обратите внимание на вывод символов " и \, для вывода которых используются управляющие последовательности: '\"' – для символа двойная кавычка и '\\' – обратная косая черта.

Задание 3

1) Описать символьный массив s без указания размера и инициализировать его строкой "Проверка!".

2) С помощью puts вывести массив s.

3) Добавить в программу определение указателя на символьный тип ptr_s и инициализировать его адресом строки "Контрольная строка".

4) Используя указатель ptr_s и функцию putchar, напечатать третий символ строки.

5) Перейти на новую строку экрана (с помощью putchar).

6) Используя указатель ptr_s и функцию puts, напечатать только второе слово ("строка").

Задания 4-6 выполнить с использованием функций. функции, их параметры и возвращаемые значения определить самостоятельно.

Задание 4.

Написать программу, которая соединяет две введенные строки (длиной не более 50 символов) в одну и печатает результат. Соединение (операция конкатенация) произвести в двух вариантах (1 строка + 2 строка и 2 строка + 1 строка).

Задание 5.

Составить программу, которая исключает из вводимой строки любой из символов, заключенных в заданной строке s2 (например, определенной как char s2[]="xyz";) и выводит измененную строку.

Задание 6.

Написать программу, вводящую символьную строку, подсчитывающую ее длину и печатающую полученный результат. Если строка длиннее заданной максимальной длины, то она "обрезается" (напечатать строку).

САМОСТОЯТЕЛЬНАЯ РАБОТА

Выполнить задания лабораторной работы №4 с использование строк (в предположении, что длина вводимой последовательности символов не превысит 100 элементов.)

1. Пусть во входном потоке находится последовательность литер (Строка символов). 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 латин-

скими 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.

2. Пусть во входном потоке находится последовательность литер (строка символов). 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).

ЛАБОРАТОРНАЯ РАБОТА №6 «РАБОТА С ОДНОМЕРНЫМИ МАССИВАМИ. МАССИВЫ И УКАЗАТЕЛИ.»

Цель: Выработка и закрепление навыков обработки элементов одномерных массивов.

ЗАДАНИЕ 1. Исследование последствий выхода индекса элемента за границу массива.

1. В указанном порядке описать в функции main следующие объекты типа int:
   - переменную n;
   - массивы x и y, каждый из трех элементов;
   - переменную i, используемую как параметр цикла.

2. Присвоить n значение -1.

3. Организовать цикл, в котором парами (x[0], y[0], затем x[1], y[1] и т.д.) вводятся значения элементов массивов x и y. Перед вводом каждого значения должно появляться предложение типа "Введите x[1]: ". Индексы элементов массива в этих сообщениях должны меняться в зависимости от шага цикла.

4. После цикла предусмотреть дополнительно ввод четвертой (лишней) пары значений x [3], y[3]. Будет ли компилятор выдавать сообщение об ошибке?

5. Запустить программу в отладочном режиме, остановившись перед выполнением ввода x[3] и y[3]. При выполнении программы вводить значения так, чтобы в результате каждый массив содержал значения 1, 2, 3.

6. В окно наблюдения Watch вывести значения n, x, y и i. Какое значение имеет параметр цикла после выхода из цикла? Пояснить.

7. Выполнить один следующий оператор программы - ввести число 4 как значение x[3]. Что изменилось в окне наблюдений?

8. Выполнить еще один шаг - ввести 4 как значение y[3]. Что изменилось в окне наблюдений?

9. Исходя из последовательности описания объектов и наблюдений результатов пунктов 7 - 8, нарисовать схему выделения памяти, используемую компилятором (в какой области среды выполнения программы будут создаваться исследуемые объекты и почему?).

ЗАДАНИЕ 2. Определение длины одномерного массива

В функции main описан (без указания количества элементов) и инициализирован массив arrDbl из значений типа double. Количество инициализаторов элементов массива может быть любым. Необходимо, чтобы выполнение программы начиналось с вывода информации о количестве элементов в массиве и (в цикле) значений всех элементов:

```
Массив arr, элементов - ...:
        arr[0]= ...
        arr[1]= ...
```

Результатом применения операции sizeof к имени массива является размер массива в байтах.

ЗАДАНИЕ 3. Использование указателей для доступа к элементам массива

1. Описать одномерный массив arr из 5 элементов типа int.
2. Вывести на экран в виде arr=...значение arr (какого оно типа?), и в цикле поочередно адреса всех элементов массива. Адрес каждого элемента выводим с новой строки в виде &arr[...]= ... Проанализировать результаты.
3. Используя для доступа к элементам массива адресную арифметику (а не квадратные скобки) в цикле присвоить элементам массива последовательно значения от 1 до 5.
4. Присвоить arr адрес среднего (третьего по счету) элемента массива. Что происходит при компиляции программы и как это можно объяснить? Строку закомментировать.
5. Добавить в программу указатель pArr на тип int. Присвоить указателю pArr адрес среднего элемента массива.
6. Рассматривая pArr как указатель на массив, вывести значения всех элементов. В каких пределах должен изменяться индекс у "массива" pArr?

ЗАДАНИЕ 4. Обработка элементов массива: определить в функции main( ) массивы: x из 10 элементов и y из 20 элементов. Поочередно добавляя в main( ) указанные ниже функции обрабо-

тать с их помощью оба массива. Вывести соответствующие результатам обработки сообщения. Результаты работы программы объяснить.

1. Описать функцию, определяющую упорядочены ли строго по возрастанию элементы целочисленного массива из 10 элементов.

2. Описать функцию, определяющую индекс первого элемента целочисленного массива из 10 элементов, значение которого равно заданному числу x. Если такого элемента в массиве нет, то считать номер равным –1.

3. 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 10 элементов, m - индекс первого отрицательного элемента этого массива либо число 0, если такого элемента в массиве нет.

4. Описать функцию, вычисляющую значение $\max(x_0+x_{n-1}; x_1+x_{n-2}; x_2+x_{n-3}; \ldots x_{(n-1)/2} + x_{n/2})$, где x_i - элементы вещественного массива x из n ($n \le 20$) элементов.

5. Описать функцию, вычисляющую значение $\min(x_0* x_1, x_1 * x_2 , x_2 * x_3,\ldots, x_{n-3} * x_{n-2}, x_{n-2} * x_{n-1})$, где x_i - элементы вещественного массива x из n элементов ($n \le 20$).

6. Описать функцию, вычисляющую значение $u_0*z_0+u_1*z_1+\ldots+ u_k*z_k$, где u_i – отрицательные элементы вещественного массива u из n ($n \le 20$) элементов, взятые в порядке их следования; z_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.

ЛАБОРАТОРНАЯ РАБОТА№7 «ОСНОВНЫЕ СВОЙСТВА УКАЗАТЕЛЕЙ»

Цель работы: Закрепить на практических примерах понимание основных особенностей указателей как объектов программы.

Отчет: Отчет должен включать для каждого задания тексты программ и описанные, в предложенной в заданиях форме, результаты работы программ.

ДЛЯ ИЗУЧЕНИЯ СВОЙСТВ УКАЗАТЕЛЕЙ НЕОБХОДИМО ПРИМЕНЯТЬ СПЕЦИФИЧЕСКИЕ ДЛЯ ЯЗЫКА СИ ОПЕРАЦИИ — ВЫЧИСЛЕНИЯ РАЗМЕРА И ПРИВЕДЕНИЯ ТИПА.

ОПЕРАЦИЯ ВЫЧИСЛЕНИЯ РАЗМЕРА SIZEOF

Операция sizeof возвращает р а з м е р своего операнда в байтах.

Результат применения оперцаии имеет специальный и определенный параллельно в нескольких заголовочных файлах (например, в stdio.h) тип size_t. В пакете BC 3.1 это синоним типа unsignedint (каким в этом случае может быть максимальный размер "измеряемого" операцией объекта?).

Операция sizeof используется в двух ситуациях.

а) Во-первых, наряду с остальными операциями языка Си, sizeof может участвовать в образовании вычисляемых выражений. С этой точки зрения она обладает приоритетом и ассоциативностью остальных унарных операций (вспомните, какими конкретно?):

sizeof 2, sizeof x, sizeof (2*x+1)

б) Во-вторых, в отличие от других операций, она может быть применена не только к объектам данных, но и к типу данных. В этом случае операнд (имя типа) обязательно должен быть заключен в круглые скобки. Результатом выполнения операции является размер объектов данного типа (очевидно, одинаковый для всех объектов).

Операция применима к любым типам:

а) базовым типам — int, float и т.п.,

б) типам, производным от базовых, например, к типу int* ("указатель на int") или массиву;

в) к структурным типам, созданным программистом (тема будет изучаться позже).

ЗАДАНИЕ 1.

1.
```
float x=-8.92;
int y=5;
printf("sizeof x=%u\n", sizeof x);
printf("sizeof y=%u\n", sizeof y);
printf("sizeofx+y=%u\n", sizeofx+y);
printf("sizeof (x+y)=%u\n", sizeof (x+y));
printf("sizeof x*2+1=%u\n", sizeof x*2+1);
printf("sizeof 2*y+1=%u\n", sizeof 2*y+1);
printf("sizeof ++y*2=%u\n", sizeof ++y*2);
printf("sizeof y--*2=%u\n", sizeof y--*2);
```

В отчете результаты работы программы оформить в виде таблицы

Выражение	Размер объекта в байтах	Порядок вычисления
. . .	. . .	. . .

В последнем столбце требуется расставить скобки в соответствии с приоритетами операций.

2. Написать программу, которая в отдельных строках выводит информацию о размере каждого из базовых типов языка (**char, short, int, long, float, double, longdouble**).

В строке выводится название типа данных и его вычисленный размер.

Примечание

Так как размеры базовых типов си стандартом языка строго нерегламентируются, то они зависят от реализации (т.е. От воли разработчика компилятора). Реальные размеры объектов часто нужны программисту при написании кода. Чтобы обеспечить использование программы на разных платформах, она должны быть универсальной и не привязанной к одному компилятору или платформе.

Применение операции **sizeof** позволяет вместо указания конкретных размеров объектов вставлять в программу выражения, вычисляющие эти размеры на этапе выполнения программы (run-time).

Размер указателя и указываемого объекта

В языке си **указатели** — это переменные, которые предназначены для хранения адресов объектов программы.

Как все остальные переменные, указатели должны быть обязательно определены, т.е. Им должны быть приписаны необходимые атрибуты.

Указатели могут быть определены или объявлены на внешнем (глобальном)или внутреннем (локальном) уровнях и иметь любой из классов памяти. Важнейшую роль в описании указателя играет тип данных.

Так как все указатели — это переменные, способные хранить всего навсего адрес только одного байта (первого из байтов, занимаемых объектом), то можно было бы предположить, что все указатели имеют один и тот же тип.

Однако для выполнения разадресации компилятор должен знать тип объекта, на который ссылается указатель. Тип данных определяет размер объекта данных, способ представления (т.е. Записи) данных в "ячейке"и набор допустимых операций с объектом. Зная только начальный адрес объекта и не зная его типа, компилятор не сможет правильно организовать обработку объекта.

Поэтому все указатели принадлежат к одному из типов, название которых образуется словосочетанием **"указатель на <тип_данных>"**.

Здесь **<тип_данных>** — это любой базовый или производный тип. Например, определение **int *px;** вводит в программу объект **px**, имеющий тип "указатель на **int**".

Так как тип, на который ссылается указатель, играет роль только на стадии компиляции (**compile-time**) при интерпретации указателя, то он никак не связан со способом размещения в памяти адресов объектов. Т.е. Тип указываемых данных запоминается компилятором отдельно. Он является внешним по отношению к значению адреса свойством. Поэтому все указатели имеют **один и тот же размер**.

Реальный размер указателя в байтах зависит от используемого для хранения объектов адресного пространства. В 16-разрядных приложениях для операционной системы ms-dos размер адресного пространства для хранения данных и кода определяется специальным параметром компилятора — моделью памяти.

Модель памяти устанавливается в пункте меню

options | compiler | codegeneration...

Например, в модели памяти **small** для данных используются

двухбайтовые адреса, а в модели **large** — четырехбайтовые.

Замечание

После изменения используемой модели памяти для получения эффекта, программу надо заново откомпилировать.

Задание 2.

1. Составить программу, которая позволяет вычислить и вывести на экран размеры типов и размеры указателей на объекты (строка на каждый тип). Операции выполнить для шести базовых типов

char, int, long, float, double, long double,

А также двух производных типов **"указатель на указатель"**, соответственно, на типы **char** и **double**.

Для вычисления размеров объектов применить к указанным типам данных операцию **sizeof**. Программу запустить дважды: в первом случае использовать модель памяти **small**, во-втором случае — модель **large**.

В отчете результаты работы программы оформить в виде таблицы:

модель памяти	**тип данных**	**размер элемента данных**	**размер указателя**
. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .

Адресная арифметика

Хотя значения указателей (адреса) по-своему смыслу являются беззнаковыми целыми числами, это особый тип данных со своими правилами адресной арифметики.

Задание 3.

Написать программу, в которой определяются переменные типов char, int, double.

Для каждой переменной программа выводит (в строку) ее адрес и значения выражений "**адрес + 2**" и "**адрес – 1**" (какой тип имеют эти значения?).

В отчете результаты привести в виде таблицы с колонками

тип данного	**adr (адрес объекта)**	**adr + 2**	**adr –1**
. . .	. . .	. . .	. . .

. . .	. . .	. . .	. . .

Полученные результаты пояснить.

Замечание

Вычисляемые здесь адресные выражения с практической точки зрения лишены смысла (адрес какого объекта мы получаем в результате вычислений?). Здесь эти выражения используются только для демонстрации правил адресной арифметики. Реальное применение адресной арифметики потребуется при выполнении второго пункта задания 4.

Операция приведения типа

Операция приведения типа является унарной. Она применяется к выражениям. Ее результат — значение операнда (выражения), преобразованное к заданному типу. Синтаксис операции приведения типа:

(<тип_данных>) выражение.

Выражение может иметь базовый или производный тип, в том числе и адресный.

Задание 4.

1. Какие результаты будут получены при выполнении следующего фрагмента программы и почему?

```
int x=8, y=3;
float z1, z2;
z1=x/y;
z2=(float)x/y;
printf("x/y=%f\n",z1);
printf("(float)x/y=%f\n",z2);
```

2. Написать программу, которая поочередно выводит в шестнадцатеричной форме значения байтов представления переменной **ul** **unsignedlongul=0x77bbccdd.**

Сообщения должны иметь вид **"байт ..., значение ..."**).

Для этого нужно:

а) определить в программе указатель **pul** соответствующего типа и присвоить ему адрес переменной **ul** (на сколько байт ссылается этот указатель при использовании среды bc++?);

б) определить указатель на один байт (какой тип он должен иметь?) И, используя операцию приведения типа, инициализировать его адресом объекта **ul**;

в) в цикле, используя правила адресной арифметики для второго указателя, вывести в отдельных строках байты шестнадцатеричного представления **ul**.

Обеспечить переносимость кода: цикл не должен зависеть от конкретного размера объектов типа **long** (использовать операцию **sizeof**). Какой особенностью обладает представление значения в памяти?

ЧАСТЬ 2

«Одномерные массивы как аргументы функций»

Цель работы: Освоить технику передачи в функции и возврата из функций массивов значений. Использование директивы **define** и статических переменных для управления работой программы.

Отчет: Отчет должен включать тексты программ из заданий 1 и 2 с необходимыми пояснениями.

ЗАДАНИЕ 1. (**Передача одномерного массива в функцию**)

1. Написать функцию **print_arr**, которая предназначена для вывода на экран значений элементов типа int из одномерного массивов любой длины. Функция выводит значения в виде таблицы. Количество элементов, выводимых в строке таблицы, задается именованной константой **COLUMNS** (если число элементов не кратно **COLUMNS**, последняя строка будет неполной).

Функция **print_arr** имеет два аргумента:

а) **arr** — массив значений типа **int**;

б) **len** — длина массива.

Функция выводит значения с указанием соответствующего элемента:

arr[0]= ... arr[1]= ... и т.д.

Значения в строке вывода разделяются с помощью символа табуляции.

З а м е ч а н и я

а) При описании аргумента-массива использовать синтаксис описания формального параметра с указателем.

б) Для отслеживания необходимости перевода строки использовать статическую переменную **count** — счетчик выведенных в строку значений.

2. В функции main описать и инициализировать массив целых значений

int a[7]={...};

С помощью вызова **print_arr** вывести значения элементов

массива на экран.

Программу запустить дважды: выводить по два и по три элемента в строку.

З а м е ч а н и е

При вызове функции размер массива должен вычисляться(!) с помощью операции **sizeof**.

3. В функции **print_arr** изменить способ описания формального параметра (массива), используя синтаксис с квадратными скобками.

а) Проверить правильность работы программы после внесенных изменений.

б) Применяя операцию **sizeof** к аргументу-массиву, вывести размер аргумента и убедиться, что внутри функции размер передаваемого массива теряется (при любом синтаксисе описания формального параметра передается указатель на первый элемент).

ЗАДАНИЕ 2 (Возврат массива из функции, продолжение программы задания 1)

1. Написать функцию **get_arr**, которая предназначена для заполнения значениями одномерных массивов произвольной длины. Функция имеет два аргумента

а) **arr** — массив значений типа **int**;

б) **len** — длина массива.

Функция выводит текст "Введите значения элементов массива:". Затем предлагает ввод элементов поочередно в виде "**arr[0]= ", "arr[1]=** " и так для всех элементов.

Замечание

Для ввода значений попробовать два варианта записи аргумента **scanf**:

а) использовать для представления адреса элемента индексное выражение (с квадратными скобками);

б) использовать для представления адреса элемента массива вычисляемое значение указателя на элемент.

2. Функция **main** теперь должна иметь следующую структуру.

Массив **a** определяется без инициализации: **int a[7];**

Вызывается функция **get_arr** и в диалоге вводятся значения элементов. Выводится текст "**Элементы массива получили значения:**".

Вызывается функция **print_arr**.

3. Размер массива в функции **main** определить с помощью именованной константы **ARRAY_LENGTH**, определяемой препроцессорной директивой:

inta[ARRAY_LENGTH];

Проверить работу программы при разных размерах массива.

ЛАБОРАТОРНАЯ РАБОТА №8
« ДВУМЕРНЫЕ МАССИВЫ, КАК АРГУМЕНТЫ ФУНКЦИЙ»

Цель работы: Освоить технику передачи в функции и возврата из функций массивов значений.

ЗАДАНИЕ 1. Возврат массива из функции.

1. Написать функцию get_arr, которая предназначена для заполнения значениями двумерных массивов произвольной длины. Функция имеет два аргумента

а) arr - указатель на первую строку массива;

б) len - число элементов в строке.

Функция выводит текст "Введите значения элементов массива:".Затем предлагает ввод элементов поочередно в виде "arr[0][0]= ","arr[0][1]= " и так для всех элементов.

2. Функция main должна иметь следующую структуру.

Массив a определяется без инициализации: int a[N1][N2];

Вызывается функция get_arr и в диалоге вводятся значения элементов.

Выводится текст "Элементы массива получили значения значения:".

Вызывается функция print_arr (см. ниже).

3. Размер массива в функции main определить с помощью именованных констант N1 и N2, определяемых препроцессорной директивой. Проверить работу программы при разных размерах массива.

ЗАДАНИЕ 2. Передача двумерного массива в функцию.

1. Написать функцию print_arr, которая предназначена для вывода на экран значений элементов типа int двумерного массивов любой длины. Функция выводит значения в виде таблицы. Количество элементов, выводимых в строке таблицы совпадает с количеством элементов в строке массива

Функция print_arr имеет два аргумента:

1. arr - указатель на первую строку массива;
2. len - число элементов в строке.

Функция выводит значения с указанием соответствующего

элемента:

arr[0][0]= ... arr[0][1]= ... и т.д.

Значения в строке разделяются с помощью символа пробела.

ЗАДАНИЕ 3. Использование адресной арифметики для обращения к элементам

двумерного массива.

1. Написать функцию print_arr_second, которая предназначена вывода на экран значений элементов типа int двумерного массивов любой длины. Функция выводит значения в виде таблицы. Количество элементов, выводимых в строке таблицы задается четвёртым параметром функции Функция print_arr имеет четыре аргумента:

1. arr - указатель на первый элемент массива;
2. len1 - число строк;
3. len2 - число элементов в строке.
4. n - количество элементов, выводимых в строке таблицы

Значения в строке разделяются с помощью символа пробела

ЛАБОРАТОРНАЯ РАБОТА №9
"ПРИЕМЫ ОБРАБОТКИ СИМВОЛЬНЫХ СТРОК"

Цель работы: Получение основных навыков, необходимых для обработки строк.

ПРЕДСТАВЛЕНИЕ СТРОК

Строка-это одномерный массив символов, заканчивающийся символом '\0', который называется нуль-терминатором. Таким образом, для строки надо резервировать место в памяти, достаточное для размещения всех символов строки и еще нулевого символа. Как известно, в Си не осуществляется автоматический контроль выхода за границы массива. Строка - особый вид массива, который имеет символ конца, что позволяет контролировать выход за границы строки.

Строка - наиболее часто используемый вид одномерного массива.

ОПРЕДЕЛЕНИЕ И ИНИЦИАЛИЗАЦИЯ СТРОК

Строки определяются как одномерный массив. Инициализировать строки можно как одномерный массив

```
charstr[7]={'с',  'т',  'р',  'о',  'к',  'а',   '\0'};
```

или строковой константой

```
charstr [7]="строка"; //символ '\0' будет добавлен автоматически
```

Со строками связаны указатели, так str -указатель на первый символ строки. Строку можно определить и через переменную-указатель. char *s="строка"; при такой инициализации указателя произойдет выделение памяти для 7 символов (почему не 6) и адресом первого символа 'с' станет значение указателя s. Указатель s является переменной, в то время как str-указатель-константа.

Неинициализированный указатель (char *s;) содержит "мусор", который будет восприниматься как адрес. Нельзя использовать его в выражении типа

```
*s='с' ; // символ 'с' будет записан по "мусорному" адресу!
```

Так как строка - массив, имеющий признак конца, программист, имеет возможность контролировать выход за границу строки.

Пример обработки строки:

```
i=0;  while   (str [i] !='\0')   (i++;...}
```

При создании строки надо следить за тем, чтобы строка заканчивалась

нуль-терминатором.

ЗАМЕЧАНИЕ

Во всех заданиях должна быть предусмотрена очистка экрана перед первым выводом, а после последнего вывода - задержка экрана пользователя (чтобы не пользоваться ALT+F5).

Задание 1.

1. Написать три функции, которые получают строку, подсчитывают и возвращают ее длину, но для работы со строкой используют разные циклы: while, for, do...while.

Написать программу, в которой вводится строка и подсчитывается ее длина, путем обращения к каждой из написанных функций. После каждого вызова функции исходная строка и ее длина выводятся на экран.

2. Написать три функции, которые получают строку и копируют ее в другую строку, используя разные циклы: while, for, do...while. Сколько и каких параметров должно быть у функций? Что должны возвращать функции?

Написать программу, в которой вводится строка и создастся ее копия, путем обращения к каждой из написанных функций. После каждого вызова функции исходная строка и ее копия выводятся на экран.

3. Написать программу, в которой вести строку, состоящую из латинских букв и цифр; подсчитать количество букв и цифр, входящих в эту строку, напечатать текст "В строке... букв и ... цифр". Проверять, является ли символ буквой или цифрой, по коду символа,

4. Написать программу, в которой ввести строку, состоящую из латинских букв и цифр; разбить исходную строку на две: в первой должны быть только буквы исходной строки, во второй - только цифры. Вывести полученные строки.

Создать новую строку в которой, вначале будут находиться все цифры из введенной строки, а затем буквы из нее же и напечатать новую строку.

Задание 2.

Написать программу, в которой

1. Ввести две символьных строки и распечатать в виде "Первая строка:..." и "Вторая строка:...".

2. Проверить, сколько первых символов в строках совпадают.

3. Напечатать количество совпадающих символов с текстом "Совпало символов:...".

4. Подсчитать количество символов, остающихся после последнего совпадения в каждой из строк и напечатать "В первой и второй строке осталось соответственно ... и ... символов ".

5. Составить новую строку, состоящую из оставшихся после последнего совпадения символов (начать символами первой строки, а продолжить из второй).

6. Составить новую строку путем чередования символов обеих исходных строк (начать с первого символа первой строки, затем первый символ второй, второй символ первой и т.д.).

Задание 3.

Написать программу, в которой

Ввести строку, содержащую десятичное число с дробной частью.

Подсчитать и вывести число цифр в целой части числа в виде "

Целая часть числа ... содержит ... цифр".

Напечатать строку в обратном порядке.

ФУНКЦИИ (МАКРОСЫ) ДЛЯ ПРОВЕРКИ ТИПА СИМВОЛЬНЫХ ЛИТЕР

Проверить, что символ с является цифрой достаточно просто

с >= '0' && с <= '9' убедиться, что символ с является русской буквой тоже несложно:

с>='А' && с<='Я' || с>='а' && с<= 'я'

Проверить, что символ с является символом верхнего регистра

или пробельным символом уже сложнее. Существуют специальные

функции (макросы) для проверки типа символьных литер, которые

собраны в библиотеке с заголовочным файлом <ctype.h>.

Аргумент каждой из них имеет тип int, если обратиться к функции с

аргументом типа char-он будет приведен к типу int

Функция isascii(c) проверяет, соответствует ли ее аргумент набору

ASCII-символов (0-127); если с является ascii-кодом символа, то возвращается ненулевое значение, в противном случае - ноль.

Аргументы остальных должны находиться в пределах 0 - 255 и быть значением типа unsignedchar. Функции проверяют для своего аргумента некоторое условие и возвращают ненулевое значение, если условие выполняется. В противном случае возвращается ноль.

Таблица функций, проверяющих тип символа

функция	проверяемое условие
isalpha(c)	с-латинская буква
isdigit(c)	с-десятичная цифра
isxdigit(c)	с- шестнадцатиричная цифра
isalnum(c)	с-латинская буква или цифра
iscntrl(c)	с-управляющий символ (0-0x1 F)
isprint(c)	с-печатаемый символ, включая пробел
ispunct(c)	с-знак пунктуации
isspace(c)	с- пробел, новая строка, табуляция
isupper(c)	с-буква верхнего регистра
islower(c)	с-буква нижнего регистра

Помимо перечисленных есть две функции, приводящие символы . к одному из регистров:

inttolower(int c) переводит с на нижний регистр;

inttoupper (int c) переводит с на верхний регистр;

Задание 4.

Написать функцию voidcheck_type(int c), которая проверяет принадлежность символа 'c' к одному или нескольким из перечисленных выше типов. Если символ принадлежит какому-то типу, то выводится соответствующеесообщение.Если типов несколько, то они перечисляются через запятую. Например: .

check_type('z');

Результат: z - буква, нижний регистр, печатаемый символ.

Написать программу, в которой ввести сим-

вольную строку, включающую латинские буквы верхнего и нижнего регистров, цифры и пробелы. Выполнить проверку типа каждого символа строки (для каждого символа - новая строка вывода).

ЛАБОРАТОРНАЯ РАБОТА № 10
« СТАНДАРТНАЯ БИБЛИОТЕКА СИ ДЛЯ РАБОТЫ С СИМВОЛЬНЫМИ СТРОКАМИ»

Цель работы: Изучить основные функции библиотеки и научиться применять их для решения задач.

Функции для работы с символьными строками собраны в библиотеке с заголовочным файлом string.h. В этом заголовочном файле определен тип size_t. Размер его может различаться на разных платформах, но в любом случае size_t является разновидностью целого без знака.

Определение длины строки

size_tstrlen(const char *s);

Функция **strlen**() возвращает длину строки, адресуемой параметром s. Символ конца строки (‘\0’) не считается.

Копированиестроки

char* strcpy(char *dest, const char *src);

Функция strcpy() копирует строку, адресуемую параметром src (от source), в область памяти, которая начинается с адреса dst (от destination). Если области src и dst перекрываются, поведение функции не определено.

Функция strcpy() возвращает указатель на строку dest.

Для копирования части строки используется функция

char *strncpy(char *dest, const char *src, size_t n);

Функция копирует не более n символов из src в dest.

Если n больше длины строки src, то “недостающие символы” забиваются нулевыми.

Если n меньше длины строки src, то нуль терминатор не добавляется. Поэтому функцию можно использовать для внесения изменений внутри строки.

Функция возвращает указатель на строку dest.

ЗАДАНИЕ 1

Написать программу, в которой

• Описать массив символов s и инициализировать его строкой "1234567890".

• Вычислить длину строки и динамически выделить память для копии строки.

• Выполнить копирование. Вывести копию строки на экран (с поясняющим текстом, что напечатана копия).

• С помощью функции strncpy заменить в копии строки три

символа, начиная со второго. Для замены вводим короткую строку с клавиатуры.

• Вывести преобразованную строку на экран.

Конкатенациястрок

char *strcat(char *dest, const char *src);

Функция strcat() дописывает копию строки src в конец строки dest. На место нуль-терминатора строки dst записывается первый символ строки src. Функция возвращает указатель на dest (на результирующую строку). Строка src при этом не изменяется.

Под dst должно быть выделено достаточно места, чтобы можно было приконкатенироватьsrc.

char *strncat(char *dest, const char *src, size_t n);

Функция strncat() дописывает не более n первых символов строки src в конец строки dest.

Задание 2

Написать программу, в которой

• С клавиатуры вводятся четыре строки разной длины (не более 10 символов). Память для каждой из них выделяется динамически.

• Необходимо объединить эти строки их в одну, память для которой выделяется в соответствии с потребностью динамически.

Сравнение строк

intstrcmp(const char *s1, const char*s2);

Функциястrcmp() выполняетсравнениестрок s1 и s2. Сравнение проводится посимвольно и последовательно, начиная с первых символов.

Функция возвращает положительное целое значение, если первая строка s1 "больше" второй s2, отрицательное, если "меньше", и нуль, если строки совпадают. Конкретное возвращаемое значение для несовпадающих строк стандартом не оговорено. Но обычно просто по порядку вычитаются коды соответствующих символов строк. Последнее вычитание проводится, когда

- коды очередных символов не равны друг другу,
- встретился конец одной строки или обеих строк (т.е. нулевой код).

intstrncmp(const char *s1, const char*s2, size_t n);

Функция strncmp() сравнивает не более n первых символов строк s1 и s2.

Задание 3

Написать программу, в которой

• ввести четыре символьные строки;

• найти среди них "наибольшую". Промежуточные результаты

выводить на экран в виде сообщения типа

<строка-1> "больше"/"меньше" <строка-2>.

•Отдельно напечатать "наибольшую" строку.

Поискподстроки

char *strstr(const char *s1, const char *s2);

Функция осуществляет поиск подстроки s2 в строке s1. Если строка s1 содержит подстроку s2, то функция возвращает указатель на первое вхождение подстроки. Если подстрока s2 в строке s1 не содержится, то функция возвращает нулевой указатель (NULL).

Задание 4

Написать программу, в которой

• Создать список фамилий в виде одной символьной строки. Например,

"1. Иванов И.И. 2. Петров П.П. 3. Сидоров С.С. 4. Александров А.А."

• В цикле в режиме диалога вводить фамилию и выводить сообщение, есть ли такой человек в списке. Признак конца диалога - пустая строка.

Разбиение строки на лексемы

char *strtok(char *s1, const char *s2);

Функция strtok() используется для разбиения строки s1 на лексемы (token). Используемые разделители лексем являются символами строки s2. Первый вызов strtok() возвращает указатель на первый символ первой лексемы в строке s1 и записывает нуль-терминатор непосредственно сразу за выделенной лексемой (на место найденного разделителя). Чтобы продолжить выделение лексем в оставшейся части строки s1, нужно повторять вызовы, задавая в качестве первого аргумента NULL пока не будут найдены все лексемы. Каждый раз функция возвращает указатель на найденную лексему. Когда достигнут конец строки, функция возвращает NULL. При повторных вызовах strtok() строку разделителей s2 можно изменять.

Задание 5

Дописать фрагмент программы, выполнить ее, пояснить текст и результаты.

```
char *ptr;
ptr=strtok("Feb. 15, 2001", "./");
printf("%s/n", ptr);
  while(ptr)
  {
ptr=strtok(NULL,".,-/");
   if(ptr) printf("%s/n", ptr);//Зачемздесьнужнапроверкаptr?
}
```

ЗАПОЛНЕНИЕ СТРОКИ СИМВОЛОМ.

char *strset(char *s, intch);

Функция strset() заполняет всю строку s символом ch (до нуль-терминатора) и возвращает указатель на s.

char *strnset(char *s, intch, int n);

Функция strnset() заполняет строку s не более чем n символами ch (не заходя за нуль-терминатор) и возвращает указатель на s.

Задание 6

Написать программу, в которой описать строку s (как в задании №1). Заменить символы с третьего по пятый на "минус", а с шестого и до конца строки - на "плюс". Вывести на экран исходную строку, промежуточный и окончательный результат.

КОПИРОВАНИЕ СТРОКИ С ВЫДЕЛЕНИЕМ ПАМЯТИ.

char* strdup(const char *str);

Функция strdup() (от stringduplicate) совмещает две операции. Она динамически выделяет память, необходимую для копии строки s, и затем копирует строку в эту область. Так как память выделяется обычной функцией malloc, то программист должен предусмотреть освобождение памяти, когда копия строки становится ненужной. Если память выделена успешно и строка скопирована, то функция возвращает указатель на копию строки. Если операцию выполнить не удалось, то функция возвращает нулевой указатель NULL.

Задание 7

Написать свою функцию str_dup(), которая работает аналогично библиотечной функции strdup().

Написать программу, в которой

- ввести строку;
- с помощью функций str_dup() и strdup() создать две копии введенной строки;
- вывести на экран две копии строки.

ЛАБОРАТОРНАЯ РАБОТА №11
СОЗДАНИЕ ПРОСТОГО УПОРЯДОЧЕННОГО ПО НЕВОЗРАСТАНИЮ СПИСКА НЕОТРИЦАТЕЛЬНЫХ ЦЕЛЫХ ЧИСЕЛ

(Например, 12->10->9->7->3->NULL)

Задание:

1. Для описания узлов списка создать структуру Node с двумя полями:
а) значение элемента списка (целое число),
б) указатель на следующий узел.

2. С помощью typedef определить обозначения NODE и pNODE соответственно для узла и указателя на него.

3. Для описания списка создать структурный тип List с двумя полями:
а) begin - указатель на первый элемент списка;
б) len - количество элементов в списке.

4. Определить синонимы LIST и pLIST для типов список и указатель на список.

5. Написать функцию создания нового (пустого) списка

pLISTcreateList(void);

При успешном создании списка функция возвращает указатель на созданную структуру типа LIST, иначе NULL.
Поля созданной структуры обнуляются.

6. Написать функцию-предикат

intisEmpty(pLISTpL);

используемую для определения, список пуст (1) или нет (0).

7. Написатьфункцию

pNODEgetPointer(pLISTpL, int date);

для поиска в упорядоченном по невозрастанию списке места для вставки узла со значением date.

Функция возвращает указатель на предшествующий найденному месту узел.

Если в список пуст - возвращается NULL.

Для узлов, стоящих па первом и втором местах, возвращается pL->begin.

8. Написатьфункцию

intaddNodeAfter(pLISTpL, pNODEpN,intnewdate);

для добавления в список нового элемента newdate после узла, на который указывает pN, исключение составляют случаи,

1) когдарN == pL->begin!=NULL;
2) когда pN ==NULL.

При успешном добавлении возвращается 1, в противном случае - 0.

9. Написатьфункцию

pNODEfindNode(pLISTpL, int date);

для поиска в списке узла со значением date. Функция возвращает указатель на предшнествующий найденному узел. Если в списке нет элемента со значением date - возвращается NULL. Для узлов, стоящих па первом и втором местах, возвррL->begin. Функция используется для удаления узла.

10. Написатьфункцию

int delNode(pLIST pL, pNODE pN);

для удаления из списка элемента, на который указывает pN.

11. Написать функцию

voidclearList(pLISTpL);

для удаления всех узлов списка (очистки списка).

12. Написать функцию

voidshowSList(pLISTpL);

которая последовательно просматривает весь список и для каждого
узла выводит на экран его адрес, находящееся число и адрес следующего
узла. В случае пустого списка выводится соответствующее сообщение.

13. Написатьфункцию

void deleteList(pLIST pL);

для удаления списка.

14. В функции main
а) создать пустой список;
б) работа со списком осуществляется в режиме диалога:
вывести меню для выбора одной из возможных операций
-добавления элемента в список (список должен быть упорядочен
по значениям целых чисел!);
-поиск числа в списке;
-удаление числа из списка;
-просмотр списка;
-очистка списка;
-конец работы.