Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

**Практическая работа №1**
по дисциплине «Информатика: Основы программирования»
на тему «Структура программы, основные типы данных, ввод/вывод»

Выполнил:
Студент Альков В.С.
Группа И407Б

Преподаватель: Першин Д.В.

Санкт-Петербург
2020 г.

**Задание 1.**

Написать программу, которая будет находить сумму любых двух целых чисел, введенных с клавиатуры.

*Входные данные:* слагаемые, два целых числа. Обозначим их a и b, тип int.

*Выходные данные:* сумма, целое число. Обозначим как s, тип int.

Таблица тестирования:

| Входные данные | Ожидаемый результат | Результат работы программы |
|---|---|---|
| a = 2, b = 2 | 4 | 4 |
| a = 2000, b = -2000 | 0 | 0 |
| a = 2000000000, b = 2000000000 | 4000000000 | -294967296 |

Текст программы:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a, b, s;          /* объявление переменных */
    printf ("a = ");      /* печать сообщения */
    scanf ("%d", &a);     /* ввод с клавиатуры вещественного числа и запись его в переменную a */
    printf ("b = ");      /* печать следующего сообщения */
    scanf ("%d", &b);     /* ввод с клавиатуры вещественного числа и запись его в переменную b */
    s = a + b;            /* вычисление суммы и запись ее в переменную s */
    printf ("%d + %d = %d\n", a, b, s); /* вывод результата в формате число + число = число */
    return 0;
}
```

*Выводы: В третьем примере результат работы программы не совпадает с ожидаемым, это происходит из-за того, что тип* int *может хранить числа в диапозоне [-2147483648,2147483647], при сложении 2000000000 + 2000000000, что в двоичном коде будет равно 01110111001101011001010000000000 + 01110111001101011001010000000000 получается 11101110011010110010100000000000, что в дополнительном коде равно результату работы программы -294967296*

**Задание 2.**

Написать программу деления одного целого числа на другое.

*Входные данные*: делимое и делитель, целые числа, обозначим их a и b, тип int.

*Выходные данные*: результат деления, целое, обозначим r, тип int.

Таблица тестирования:

| Входные данные | Ожидаемый результат | Результат работы программы |
|---|---|---|
| a = 10, b = 2 | 5 | 5 |
| a = 5, b = 2 | 2.5 | 2 |
| a = 1, b = 2 | 0.5 | 0 |

Текст программы:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a, b, r;         /* объявление переменных */
    printf ("a = ");     /* печать сообщения */
    scanf ("%d", &a);    /* ввод с клавиатуры вещественного числа и запись его в
переменную a */
    printf ("b = ");     /* печать следующего сообщения */
    scanf ("%d", &b);    /* ввод с клавиатуры вещественного числа и запись его в
переменную b */
    r = a / b;           /* вычисление результата деления и запись его в переменную
r */
    printf ("%d / %d = %d\n", a, b, r); /* вывод результата в формате число /
число = число */
    return 0;
}
```

Выводы: при делении целого на целое результат остается целым, дробная часть отбрасывается, поэтому когда число не делится нацело результат работы программы не совпадает с ожидаемым.

**Задание 3.**

Изменить тип переменных в предыдущей программе на *double* (стандартный вещественный тип). В функциях *scanf()* и *printf()* поменять спецификаторы формата на *%lf*. *Входные* и *выходные данные* те же, что и в задании 2, обозначения переменных те же, тип всех переменных double.

Таблица тестирования:

| Входные данные | Ожидаемый результат | Результат работы программы |
|---|---|---|
| a = 10, b = 2 | 5 | 5.000000 |
| a = 5, b = 2 | 2.5 | 2.500000 |
| a = 1, b = 2 | 0.5 | 0.500000 |

Текст программы:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    double a, b, r;         /* объявление переменных */
    printf ("a = ");     /* печать сообщения */
    scanf ("%lf", &a);    /* ввод с клавиатуры вещественного числа и запись его в
переменную a */
    printf ("b = ");     /* печать следующего сообщения */
    scanf ("%lf", &b);    /* ввод с клавиатуры вещественного числа и запись его в
переменную b */
    r = a / b;           /* вычисление результата деления и запись его в переменную
r */
    printf ("%lf / %lf = %lf\n", a, b, r); /* вывод результата в формате число /
число = число */
    return 0;
}
```

Выводы: если задействован тип double, дробная часть всегда будет у переменной, независимо от того значима она или нет.

При изменении формата вывода на %.8lf выводимое значение стало таким: 5.00000000, 2.50000000, 0.50000000.

При изменении формата вывода на %.2lf выводимое значение стало таким: 5.00, 2.50, 0.50.

Выводы: если изменить формат вывода на %.числоlf, можно задать кол-во символов для дробной части(точность), но кол-во возможных значащих для выбранного типа не измениться, то что за пределами типа будет дополнено нулями.

**Задание 4.**

В предыдущей программе изменить тип делимого и делителя обратно на int, результат оставить типа double.

*Входные данные*: делимое и делитель, целые, обозначим их a и b, тип int.

*Выходные данные*: результат деления, число с плавающей точкой, тип double.

Таблица тестирования:

| Входные данные | Ожидаемый результат | Результат работы программы |
|---|---|---|
| a = 10, b = 2 | 5 | 5.000000 |
| a = 5, b = 2 | 2.5 | 2.500000 |
| a = 1, b = 2 | 0.5 | 0.000000 |

Текст программы:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a, b;           /* объявление переменных */
    double r;
    printf ("a = ");    /* печать сообщения */
    scanf ("%d", &a);   /* ввод с клавиатуры вещественного числа и запись его в
переменную a */
    printf ("b = ");    /* печать следующего сообщения */
    scanf ("%d", &b);   /* ввод с клавиатуры вещественного числа и запись его в
переменную b */
    r = a / b;          /* вычисление результата деления и запись его в переменную
r */
    printf ("%d / %d = %lf\n", a, b, r); /* вывод результата в формате число /
число = число */
    return 0;
}
```

Выводы: происходит деление целого на целое, в таком случает дробная часть отбрасывается и значение деления приводится к типу double.

**Задание 5.**

Проанализировать ошибки при вызове функций scanf() и printf().

| Ошибка | Поведение программы |
|---|---|
| отсутствие & перед именем переменной в scanf() | Переменной не будет присвоено значение. &переменная = адрес ячейки памяти, куда нужно присвоить значение, если |

| | |
|---|---|
| | & не будет, то значение будет присваиваться неопределенной ячейке памяти, с номером которой совпадает значение переменной. |
| наличие & перед именем переменной в printf() при выводе значения переменной | Будет выведен номер ячейки памяти переменной. |
| тип спецификатора формата ввода не совпадает с типом переменной: переменная типа *int*, спецификатор *%lf* | Переменная обнуляется |
| тип спецификатора формата ввода не совпадает с типом переменной: переменная типа *double*, спецификатор *%d* | Если выводит такую переменную со спецификатором вывода %lf, то выведется 0, а если со спецификатором вывода %d, то выведется целая часть числа. |
| тип спецификатора формата ввода не совпадает с типом переменной: переменная типа *double*, спецификатор *%f* | со спецификатором %lf выводится 0. |
| тип спецификатора формата вывода не совпадает с типом значения: значение типа *int*, спецификатор *%lf* | Программа вывела 0.000000, мне кажется, что это связано с разными методами хранения чисел в памяти, программа собирается раскодировать число методом, созданным для чисел с плавающей точкой, а число закодировано по-другому. |
| тип спецификатора формата вывода не совпадает с типом значения: значение типа *double*, спецификатор *%d* | Выводиться неверное значение. Мне кажется, что это связано с разными методами хранения чисел в памяти, программа раскодировала число способом для целых. |
| количество спецификаторов формата ввода меньше количества вводимых значений переменных | В программу вводиться только одно число. |
| количество спецификаторов формата ввода больше количества вводимых значений переменных | Программа все равно просить ввести число, неизвестно куда число попадает. |
| количество спецификаторов формата вывода меньше количества выводимых значений | Программа выводит столько значений, сколько спецификаторов вывода. |
| количество спецификаторов формата вывода больше количества выводимых значений | Программа выводит все значения, одно – значение переменной, другие неопределенные, возможно из каких-то ячеек памяти. |

**Задание 6.**

Познакомьтесь с типами данных *char* и *unsigned char*.

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>     /* библиотека, содержащая хар-ки типов переменных */

int main()
{
```

```
    char c;              /* объявление переменных */
    unsigned char uc;
    /* sizeof() - функия возвращающая байтность типа переменной */
    printf("sizeof(c)=%d\tsizeof(uc)=%d\n\n", sizeof(c),sizeof(uc));
    uc = c = CHAR_MAX;  /* CHAR_MAX = 01111111, c = 127 uc = 127 */
    /* вывод сообщения в формате CHAR_MAX : c= переменная_c uc=переменная_uc */
    printf("CHAR_MAX : c=%d uc=%d\n", c, uc);
    /* CHAR_MAX+1 = 10000000, 128 = 10000000, -128 = 01111111+1=10000000
    (в дополнительном), знаковая переменная c=-128, так как знаковый бит
    определяет знак   числа, uc = 128, так как тип беззнаковый, и знаковый бит
    используется не для знака.*/
    c = c + 1; uc = uc + 1;
    /* вывод сообщения в формате CHAR_MAX+1 : c= переменная_c uc=переменная_uc */
    printf("CHAR_MAX+1 : c=%d uc=%d\n", c, uc);
    /* CHAR_MIN = 10000000 = -128 в дополнительном, 128 в прямом, знаковая
    переменная c=-128, так как знаковый бит определяет знак числа, uc = 128, так
    как тип беззнаковый, и знаковый бит используется не для знака.*/
    uc = c = CHAR_MIN;
    /* вывод сообщения в формате CHAR_MIN : c= переменная_c uc=переменная_uc */
    printf("CHAR_MIN : c=%d uc=%d\n", c, uc);
    /* UCHAR_MAX = 11111111 = 255 в прямом, -1 в дополнительном, uc=255, так как
    тип беззнаковый, и знаковый бит используется не для определения знака, можно
    сказать, что число кодируется в прямом коде, c = -1, так как тип знаковый,
    знаковый бит определяет знак числа, можно сказать, что кодируется в
    дополнительном коде. */
    c = uc = UCHAR_MAX;
    /* вывод сообщения в формате UCHAR_MAX : c= переменная_c uc=переменная_uc */
    printf("UCHAR_MAX : c=%d uc=%d\n", c, uc);
    /* UCHAR_MAX+1 = 11111111 + 1 = 00000000 = 0, используется дополнительный код,
    в нем при переполнении старший бит стирается, uc = 0, c = 0*/
    c = c + 1; uc = uc + 1;
    /* вывод сообщения в формате UCHAR_MAX+1 : c= переменная_c uc=переменная_uc */
    printf("UCHAR_MAX+1 : c=%d uc=%d\n", c, uc);
    /* c = -5 = 11111011, так как тип знаковый, знаковый бит определяет знак
    числа, можно сказать, что кодируется в дополнительном коде.
    uc = 11111011 = 251, так как тип беззнаковый, и знаковый бит используется
    не для определения знака, можно сказать, что число кодируется в прямом коде */
    uc = c = -5;
    /* вывод сообщения в формате -5 : c= переменная_c uc=переменная_uc */
    printf("-5 : c=%d uc=%d\n", c, uc);
    /* c = -5 = 11111011, uc = 5 = 00000101 */
    c = -5; uc = 5;
    /* происходит неявное приведение типа переменных c и uc к int, котороый
    знаковый, поэтому результат 0,  -5 не больше 5  */
    printf("char and unsigned char -5>5 : %d\n\n", c>uc);
    return 0;
}
```

Результаты работы программы:

```
sizeof(c)=1     sizeof(uc)=1

CHAR_MAX : c=127 uc=127
CHAR_MAX+1 : c=-128 uc=128
CHAR_MIN : c=-128 uc=128
UCHAR_MAX : c=-1 uc=255
UCHAR_MAX+1 : c=0 uc=0
-5 : c=-5 uc=251
char and unsigned char -5>5 : 0


Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```

**Задание 7.**

Познакомьтесь с типами данных *int, short int, long int* и *unsigned int*.

```
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>

int main()
{
    char c;
    unsigned char uc;
    int i;
    unsigned u;
    short s;
    long l;
    /* sizeof() - фунКия возвращающая байтность типа переменной */
    printf("sizeof(i)=%d\tsizeof(u)=%d\tsizeof(s)=%d\tsizeof(l)=%d\n\n",
          sizeof(i), sizeof(u), sizeof(s), sizeof(l));
    /* c = s = 32767 = 0111111111111111, s=32767, a c = 11111111 = -1,
    так как char вмещает только 8 битов, первые 8 отбрасываются, тип знаковый */
    c = s = SHRT_MAX;
    /* uc = s = 0111111111111111, uc = 11111111 = 255,
    так как char вмещает только 8 битов,
    первые 8 отбрасываются,  тип беззнаковый*/
    uc = s;
    printf("SHRT_MAX : c=%d uc=%d s=%d\n", c, uc, s);
    /* s = 32767 + 1 = 0111111111111111 + 1 = 1000000000000000 = -32768,
    так как тип знаковый */
    s = s + 1;
    printf("SHRT_MAX+1 : s=%d\n", s);
    /* c = 1000000000000000 = 00000000 = 0, так как char вмещает только 8 битов,
    первые 8 отбрасываются, uc = 1000000000000000 = 00000000 = 0,
    так как char вмещает только 8 битов, первые 8 отбрасываются*/
    c = s; uc = s;
    printf("%d : c=%d uc=%d\n", SHRT_MIN, c, uc);
    /* s = 0000000000000000 = 0, c= 0000000000000000 = 00000000 = 0,
    так как char вмещает только 8 битов, первые 8 отбрасываются,
    uc = 0000000000000000 = 00000000 = 0,
    так как char вмещает только 8 битов, первые 8 отбрасываются*/
    s = 0; c = s; uc = s;
    printf("0 : c=%d uc=%d s=%d\n", c, uc, s);
    /* i = 2147483647 = 01111111111111111111111111111111 */
    i = INT_MAX;
    /* l = 2147483647 = 01111111111111111111111111111111,
    u = 2147483647 = 01111111111111111111111111111111 */
    l = i; u = i;
    printf("INT_MAX : i=%d u=%u l=%ld\n", i, u, l);
    /*l = i = 01111111111111111111111111111111 + 1
    = 10000000000000000000000000000000 = -2147483648,
    так как типы знаковые и четырехбайтные,
    u = 01111111111111111111111111111111 + 1 = 2147483648,
    так как тип беззнаковый*/
    i = i + 1; l = l + 1; u = u + 1;
    printf("INT_MAX+1 : i=%d u=%u l=%ld\n", i, u, l);
    /* i = -2147483648 = 10000000000000000000000000000000*/
    i = INT_MIN;
    /* 1 = 10000000000000000000000000000000 = -2147483648,
    так как тип знаковый,
    u = 10000000000000000000000000000000 = 2147483648,
    так как тип беззнаковый*/
    l = i; u = i;
    printf("INT_MIN : i=%d u=%u l=%ld\n", i, u, l);
    /* u = 4294967295 = 11111111111111111111111111111111,
    тип беззнаковый */
    u = UINT_MAX;
```

```
    /* i = 11111111111111111111111111111111 = -1,
    так как тип знаковый,
    l = 11111111111111111111111111111111 = -1,
    так как тип знаковый и четырехбайтный */
    i = u; l = u;
    printf("UINT_MAX : i=%d u=%u l=%ld\n", i, u, l);
    /* u = i = 11111111111111111111111111111011,
    i = -5, так как тип знаковый,
    u = 4294967291, так как тип беззнаковый*/
    u = i = -5;
    printf("-5 : i=%d u=%u\n", i, u);
    /* i = 11111111111111111111111111111011,
    u = 00000000000000000000000000000101,
    i>u = 1, так как происходит неявное привидение типа
    к unsigned int, то есть сравнивается 4294967291 > 5,
    поэтому истина*/
    i = -5; u = 5;
    printf("int and unsigned int -5>5 : %d\n", i > u);
    c = -5; u = 5;      /* c = -5 = 11111011,
    u = 5 = 00000000000000000000000000000101,
    неявное приведение типа c к
    unsigned int, c = 00000000000000000000000011111011 = 251,
    то есть сравнивается 251>5, поэтому истина*/
    printf("char and unsigned int -5>5 : %d\n\n", c > u);
    /* тип переменной i - int, целочисленный,
    дробная часть отбрасывается */
    i = 5.1;
    printf("i=5.1 : i=%d\n", i);
    /* тип переменной i - int,
    целочисленный, дробная часть отбрасывается*/
    i = 5.9;
    printf("i=5.9 : i=%d\n", i);
    return 0;
}
```

Результаты работы программы:

```
sizeof(i)=4     sizeof(u)=4     sizeof(s)=2     sizeof(l)=4

SHRT_MAX : c=-1 uc=255 s=32767
SHRT_MAX+1 : s=-32768
-32768 : c=0 uc=0
0 : c=0 uc=0 s=0
INT_MAX : i=2147483647 u=2147483647 l=2147483647
INT_MAX+1 : i=-2147483648 u=2147483648 l=-2147483648
INT_MIN : i=-2147483648 u=2147483648 l=-2147483648
UINT_MAX : i=-1 u=4294967295 l=-1
-5 : i=-5 u=4294967291
int and unsigned int -5>5 : 1
char and unsigned int -5>5 : 1

i=5.1 : i=5
i=5.9 : i=5

Process returned 0 (0x0)   execution time : 0.321 s
Press any key to continue.
```

**Задание 8.**

Познакомьтесь с типами данных *float* и *double*.

```
#include <stdlib.h>
#include <stdio.h>
#include <float.h>      /* библиотека, которая содержит хар-ки типов с плавающей
точкой */

int main()
```

```
{
    float f;
    double d;
    /* sizeof() - функия возвращающая байтность типа переменной */
    printf("sizeof(f)=%d\tsizeof(d)=%d\n\n", sizeof(f), sizeof(d));
    /* FLT_MAX - const, обозн. максимальное значение,
    которое может быть представленно типом float,
    по идеи в двоичном должно выглядеть так: 01111111011111111111111111111111,
    или в шестнадцатиричной: 7F7FFFFF,
    в тип double оно помещается и выглядит так:
    0100011111101111111111111111111111100000000000000000000000000000,
    шестнадцатиричной:47EFFFFFE0000000 */
    d = f = FLT_MAX;
    printf("FLT_MAX : f=%g d=%g\n", f, d);
    /* FLT_MIN - const, обозн. минимальное нормализованное число (близкое к 0),
    которое может представить тип float */
    d = f = FLT_MIN;
    printf("FLT_MIN : f=%g d=%g\n", f, d);
    /* FLT_EPSILON - const, обозн. минимальное положительное x такое, что 1.0 + x
    != 1.0 */
    d = f = FLT_EPSILON;
    printf("FLT_EPSILON : f=%g d=%g\n", f, d);
    /* 1e10 = 10010101000000101111100010000000000,
    порядок 33 = 10100000,
    мантисса = 00101010000001011111100010000000000
    (длина 35, а может быть только 23, обрезаем)
    =  00101010000001011111001, знаковый бит = 0,
    получается такое число 01010000000101010000001011111001,
    и если перевести обратно получиться 1e10, потому что ничего не потеряли.*/
    f = 1e10;
    printf("1e10 : f=%f\n", f);
    /* 1e11 = 10111010010000111011011110100000000000,
    порядок = 33 = 10100011,
    мантисса = 01110100100001110110111010000000000
    (длина 36, а может быть только 23, обрезаем,
    тут и происходит потеря данных) = 01110100100001110110111,
    знаковый бит = 0,
    получается такое число 01010001101110100100001110110111,
    и если перевести обратно получится 99999997952*/
    f = 1e11;     /* комментарии */
    printf("1e11 : f=%f\n", f);
    /* 1234567890 = 10010011001011000000001011010010
    порядок = 30 = 10011101,
    мантисса = 00100110010110000000101,
    знак = 0,
    получается = 01001110100100110010110000000101, но если перевести обратно
    = 1234567808, видимо произошло округление последнего бита, и в памяти чилсло
    хранится так 01001110100100110010110000000110 = 1234567936
    */
    f = 1234567890;
    printf("1234567890 : f=%f\n", f);
    /* DBL_MAX - const, обозн. максимальное значение,
    которое может быть представленно типом double */
    d = DBL_MAX;
    printf("DBL_MAX : d=%g\n", d);
    /* DBL_MIN - const, обозн. минимальное нормализованное число (близкое к 0),
    которое может представить тип double */
    d = DBL_MIN;
    printf("DBL_MIN : d=%g\n", d);
    /* DBL_EPSILON - const, обозн. положительное x такое, что 1.0 + x != 1.0 */
    d = DBL_EPSILON;
    printf("DBL_EPSILON : d=%g\n", d);
    /* 1000000000000001 = 1110001101011111101010010011000110100000000000000001,
```

```
    порядок = 49 = 10000110000,
    мантисса = 11000110101111110101001001100011010000000000000001000,
    знак = 0,
    число = 0100001100001100011010111111010100100110001101000000000000000001000 =
    1000000000000001 */
    d = 1e15 + 1;
    printf("1e15+1 : d=%lf\n", d);
    /* 10000000000000001 = 10001110000110111100100110111111000001000000000000000001,
    порядок = 53 = 10000110100,
    мантисса = 00011100001101111001001101111110000010000000000000000 (последний бит
    вышел за пределы),
    знак = 0,
    число = 0100001101000001110000110111100100110111111000001000000000000000 =
    1e16 */
    d = 1e16 + 1;
    printf("1e16+1 : d=%lf\n", d);
    f = 10000 * 100000;
    printf("1 : f=%f\n", f);
    /*
    01001110011011100110101100101000 + 00111111100000000000000000000000 =
    1.11011100110101100101000 * 2^29 + 1.00000000000000000000000 * 2^0 =
    1.11011100110101100101000 * 2^29 + 0.00000000000000000000000000001 *2^29 =
    1.11011100110101100101000000001 * 2^29 =
    01001110011011100110101100101000, значение не поменялось, так как длинна
    мантиссы не может превышать 23, лишнее обрезалось */
    f += 1;
    /*
    01001110011011100110101100101000 - 01001110011011100110101100101000 =
    1.11011100110101100101000 * 2^29 - 1.11011100110101100101000 * 2^29 = 0
    */
    f -= 4 * 250000000;
    printf("1 : f=%f\n", f);
    /* f=1, потому что сперва посчитается выражение в типе int, а потом
    уже приведется к типу float, 1 не потеряется как в прошлый раз */
    f = 10000 * 100000 + 1 - 4 * 250000000;
    printf("1 : f=%f\n", f);
    /* d=1, потому что сперва посчитается выражение в типе int, а потом
    уже приведется к типу double, 1 не потеряется как в прошлый раз */
    d = 10000 * 100000 + 1 - 4 * 250000000;
    printf("1 : d=%lf\n", d);
    /* видимо так происходит, потому что значение очень большое,
    1000 на него уже никак не влияет, поэтому при вычитании
    такого же больше переменная обращается в ноль*/
    d = 1e20 * 1e20 + 1000 - 1e22 * 1e18;
    printf("1000 : d=%lf\n", d);
    /* Сперва взаимоуничтожатся большии значения, а потом к нулю прибавится 1000*/
    d = 1e20 * 1e20 - 1e22 * 1e18 + 1000;
    printf("1000 : d=%lf\n", d);
    return 0;
}
```

Результаты работы программы:

```
sizeof(f)=4     sizeof(d)=8

FLT_MAX : f=3.40282e+038 d=3.40282e+038
FLT_MIN : f=1.17549e-038 d=1.17549e-038
FLT_EPSILON : f=1.19209e-007 d=1.19209e-007
1e10 : f=10000000000.000000
1e11 : f=99999997952.000000
1234567890 : f=1234567936.000000
DBL_MAX : d=1.79769e+308
DBL_MIN : d=2.22507e-308
DBL_EPSILON : d=2.22045e-016
1e15+1 : d=1000000000000001.000000
1e16+1 : d=10000000000000000.000000
1 : f=1000000000.000000
1 : f=0.000000
1 : f=1.000000
1 : d=1.000000
1000 : d=0.000000
1000 : d=1000.000000

Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

**Задание 9.**

Проверить порядок выполнения операций в каждом выражении, содержащем несколько операций присваивания, разделив каждый оператор-выражение на несколько операторов, выполняемых последовательно.

Текст измененной программы:

```
#include <stdio.h>
#include <stdlib.h>
int main (void)
{
	int a, b = 5, c;
	double x, y = -.5, z;
	printf("a=");
	scanf("%d", &a);
	c=a;
	x=c;
	printf("x = c = a : a=%d c=%d x=%f\n",a, c, x);
	a = a + b;
	printf("a += b : a=%d\n", a);
	x = x *(b+a);
	printf("x *= b+a : x=%lf\n", x);
	b = b + a;
	a--;
	printf("b += a-- : a=%d b=%d\n", a, b);
	c++;
	x = x - c;
	printf("x -= ++c : c=%d x=%lf\n", c, x);
	c = a/b;
	printf("c = a/b : c=%4d\n",c);
	c = a%b;
	printf("c = a%%b : c=%d\n",c);
	y = y + (a+1)/a;
	a++;
	printf("y += (a+1)/a++ : a=%d y=%.3lf\ty=%.0lf\n",
	a, y, y);
	y = y - .6;
```

```
	b = 3*y+2*b+1;
	printf("b = 3*(y-=.6)+2*b+1 : b=%d y=%.1lf\n",
	b, y);
	z = a/2;
	printf("z = a/2 : z = a/2 : z=%lf\n", z);
	z = (double)a/2;
	printf("z = (double)a/2 : z=%lf\n", z);
	x = 5.7;
	y = x/2;
	printf("y = (x = 5.7)/2 : x=%lf y=%lf\n", x, y);
	y = (int)x/2;
	printf("y = (int)x/2 : y=%f\n", y);
	c = c / 2;
	z = (b-3)/2 - x/5 + c + 1/4*z - y;
	b++;
	z = z + b/3.;
	y++;
	printf("z = (b-3)/2 - x/5 + c + 1/4*z - y++ \
	+ ++b/3. :\na=%d b=%d c=%d x=%lf y=%lf z=%lf\n",
	a,b,c,x,y,z);
	system("pause");
	return 0;
}
```

Результаты работы программ:

до изменения

```
a=15
x = c = a : a=15 c=15 x=15.000000
a += b : a=20
x *= b+a : x=375.000000
b += a-- : a=19 b=25
x -= ++c : c=16 x=359.000000
c = a/b : c=   0
c = a%b : c=19
y += (a+1)/a++ : a=20 y=0.500   y=1
b = 3*(y-=.6)+2*b+1 : b=50 y=-0.1
z = a/2 : z = a/2 : z=10.000000
z = (double)a/2 : z=10.000000
y = (x = 5.7)/2 : x=5.700000 y=2.850000
y = (int)x/2 : y=2.000000
z = (b-3)/2 - x/5 +(c/=2) + 1/4*z - y++ + ++b/3. :
a=20 b=51 c=9 x=5.700000 y=3.000000 z=45.860000
Для продолжения нажмите любую клавишу . . .
```

после изменения

```
a=15
x = c = a : a=15 c=15 x=15.000000
a += b : a=20
x *= b+a : x=375.000000
b += a-- : a=19 b=25
x -= ++c : c=16 x=359.000000
c = a/b : c=   0
c = a%b : c=19
y += (a+1)/a++ : a=20 y=0.500   y=1
b = 3*(y-=.6)+2*b+1 : b=50 y=-0.1
z = a/2 : z = a/2 : z=10.000000
z = (double)a/2 : z=10.000000
y = (x = 5.7)/2 : x=5.700000 y=2.850000
y = (int)x/2 : y=2.000000
z = (b-3)/2 - x/5 + c + 1/4*z - y++ + ++b/3. :
a=20 b=51 c=9 x=5.700000 y=3.000000 z=45.860000
Для продолжения нажмите любую клавишу . . .
```

**Задание 10.**

Написать программу для вычисления значений следующих выражений:

a=5, c=5

a=a+b-2

c=c+1, d=c-a+d

a=a*c, c=c-1

a=a/10, c=c/2, b=b-1, d=d*(c+b+a)

Выражения, записанные в одной строке, записывать одним оператором-выражением, не содержащим запятой. Использовать расширенные операции присваивания, операции инкремента и декремента. Переменные c и d объявить как целые, переменные a и b – как вещественные. Значения переменных b и d вводить с клавиатуры. После вычисления каждого выражения выводить на экран значения всех переменных.

Таблица тестирования:

| Входные данные | Ожидаемый результат | Результат работы программы |
|---|---|---|
| b=1.5 d=2 | a=2.7 b=0.5 c=2 d=5 | a=2.7 b=0.5 c=2 d=5 |
| b=0.2 d=10 | a=1.92 b= -0.8 c=2 d=6 | a=1.92 b= -0.8 c=2 d=6 |
| b=3.3 d=7 | a=3.78 b=2.3 c=2 d=0 | a=3.78 b=2.3 c=2 d=0 |

Текст программы:

```
#include <stdio.h>
int main(void)
{
	int c,d;
	double a,b;
	scanf("%lf%lf",&b,&d);
	c=a=5;
	printf("a=%lf b=%lf c=%d d=%d\n",a,b,c,d);
	a+=b-2;
	printf("a=%lf b=%lf c=%d d=%d\n",a,b,c,d);
	d=++c-a+d;
```

```
	printf("a=%lf b=%lf c=%d d=%d\n",a,b,c,d);
	a*=c--;
	printf("a=%lf b=%lf c=%d d=%d\n",a,b,c,d);
	d*= (c/=2) + --b + (a/=10);
	printf("a=%lf b=%lf c=%d d=%d\n",a,b,c,d);
	return 0;
}
```