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

Кафедра O7 «Информационные системы и программная инженерия»

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

Выполнил:  
Студент *Усов Д.А..*  
Группа *Е123Б*  
  
Преподаватель:  
*Лестенко Н.А.*

Санкт-Петербург  
2022 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** |
| а = 2, b = 2 | 4 | 4 |
| а = 2000, b = -2000 | 0 | 0 |
| а = 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 целого числа и запись его в переменную a \*/

printf ("b = "); /\* печать следующего сообщения \*/

scanf ("%d", &b); /\* ввод с клавиатуры целого 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*: *первые два результата работы программы оказались такими же, как и ожидаемые результаты, в отличие от третьего. Это связано с переполнением. Для того чтобы складывать такие большие числа можно воспользоваться unsigned int*.

**Задание 2.**

Написать программу деления одного целого числа на другое.

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

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

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| a = 4, b = 2 | 2 | 2 |
| a = 7, b = 3 | 2,33 | 2 |
| a = 5, b = 3 | 1,67 | 1 |
| a = 1, b = 2 | 0,5 | 0 |
| a = 5, b = 0 | Сообщение об ошибке | Сообщение об ошибке |

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 целого числа и запись его в переменную a \*/

printf ("b = "); /\* печать следующего сообщения \*/

scanf ("%d", &b); /\* ввод с клавиатуры целого числа и запись его в переменную b \*/

s = a / b; /\* вычисление частного и запись его в переменную s \*/

printf ("%d / %d = %d\n", a, b, s); /\* вывод результата в формате число / число = число \*/

return 0;

}

Выводы: при делении одного целого числа на другое целое число, в результате получается только целое число, при делении на 0 выдается ошибка.

**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 = 4, b = 2 | 2 | 2.000000 |
| a = 7, b = 3 | 2,33 | 2.333333 |
| a = 5, b = 3 | 1,67 | 1.666667 |
| a = 1, b = 2 | 0,5 | 0.500000 |
| a = 5, b = 0 | Сообщение об ошибке | inf |
| a = 4.2, b = 2.1 | 2 | 2.000000 |
| a = 5.5, b = 2.2 | 2,5 | 2.500000 |
| a = 4.4, b = 0.1 | 44 | 44.000000 |

Текст программы:

#include <stdio.h>  
#include <stdlib.h>  
  
int main()  
{  
 double a, b, s; /\* 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 \*/  
 s = a / b; /\* вычисление частного и запись его в переменную s \*/  
 printf ("%lf / %lf = %lf\n", a, b, s); /\* вывод результата в формате число / число = число \*/  
 return 0;  
}

Выводы: при делении одного вещественного числа на другое вещественное число, в результате получается только вещественное число число, при делении на 0 выдается бесконечность.

При изменении формата вывода на *%.8lf* выводимое значение стало таким: 5.00000000 / 3.00000000 = 1.66666667.

При изменении формата вывода на %.2lfвыводимое значение стало таким: 5.00 / 3.00 = 1.67.

Выводы: при изменении формата, изменилось количество символов после точки.

**Задание 4.**

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

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

*Выходные данные*: частное, вещественное число. Обозначим как s, тип double.

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| a = 4, b = 2 | 2 | 2.000000 |
| a = 7, b = 3 | 2,33 | 2.000000 |
| a = 5, b = 3 | 1,67 | 1.000000 |
| a = 1, b = 2 | 0,5 | 0.000000 |
| a = 5, b = 0 | Сообщение об ошибке | Сообщение об ошибке |

Текст программы:

#include <stdio.h>  
#include <stdlib.h>  
  
int main()  
{  
 int a, b; double s; /\* объявление переменных \*/  
 printf ("a = "); /\* печать сообщения \*/  
 scanf ("%d", &a); /\* ввод с клавиатуры целого числа и запись его в переменную a \*/  
 printf ("b = "); /\* печать следующего сообщения \*/  
 scanf ("%d", &b); /\* ввод с клавиатуры целого числа и запись его в переменную b \*/  
 s = a / b; /\* вычисление частного и запись его в переменную s \*/  
 printf ("%d / %d = %lf \n", a, b, s); /\* вывод результата в формате число / число = число \*/  
 return 0;  
}

Выводы: при делении одного целого числа на другое целое число, в результате получается только целое число, которое записывается в переменную вещественного типа без дробной части. При делении на 0 выдается ошибка.

**Задание 5.**

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

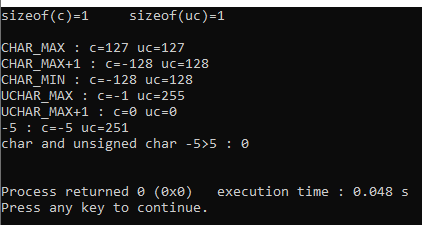
|  |  |
| --- | --- |
| **Ошибка** | **Поведение программы** |
| отсутствие & перед именем переменной в scanf() | Программа завершается с ошибкой 0xC0000005 |
| наличие & перед именем переменной в printf() при выводе значения переменной | Не выводит значение переменной, вместо этого выводит числовое значение адреса переменной в памяти |
| тип спецификатора формата ввода не совпадает с типом переменной: переменная типа *int*, спецификатор *%lf* | Программа выводит другое значение, не то, которое вводилось ранее, не совпадает тип |
| тип спецификатора формата ввода не совпадает с типом переменной: переменная типа *double*, спецификатор *%d* | Никаких проблем не возникает, выводится вводимое значение, не совпадает тип |
| тип спецификатора формата ввода не совпадает с типом переменной: переменная типа *double*, спецификатор *%f* | Программа выводит другое значение, не то, которое вводилось ранее, не совпадает тип |
| тип спецификатора формата вывода не совпадает с типом значения: значение типа *int*, спецификатор *%lf* | Программа выводит значение 0.000000, не совпадает тип |
| тип спецификатора формата вывода не совпадает с типом значения: значение типа *double*, спецификатор *%d* | При попытке вывести число с дробной частью равной 0, выводится 0. При попытке вывода числа с дробной частью не равной 0, выводится большое значение |
| количество спецификаторов формата ввода меньше количества вводимых значений переменных | Меняется значение только первой переменной, значение второй переменной остается неизменным |
| количество спецификаторов формата ввода больше количества вводимых значений переменных | Программа завершается с ошибкой 0xC0000005 |
| количество спецификаторов формата вывода меньше количества выводимых значений | Выводится столько значений, сколько указано спецификаторов. Вывод без ошибок |
| количество спецификаторов формата вывода больше количества выводимых значений | Выводится ровно столько значений, сколько указано переменных, далее будут выведены нули (0.000000) |

**Задание 6.**

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

#include <stdlib.h>  
#include <stdio.h>  
#include <limits.h> //Включение определений характеристик общих типов переменных  
  
int main()  
{  
 signed char c; //-128...127 объявление переменных  
 unsigned char uc; //0...255  
 printf("sizeof(c)=%d\tsizeof(uc)=%d\n\n", sizeof(c),sizeof(uc)); // выводим количество занимаемого места в байтах в памяти каждой переменной  
 uc=c=CHAR\_MAX; //CHAR\_MAX = 127 01111111 присваиваем это значение переменным uc и c  
 printf("CHAR\_MAX : c=%d uc=%d\n", c, uc); //выводим значения переменных uc и с  
 c = c + 1; uc = uc + 1; //01111111 + 1 = 10000000 прибавляем единицу к значениям переменных  
 printf("CHAR\_MAX+1 : c=%d uc=%d\n", c, uc); //выводим значения переменных uc и с  
 uc = c = CHAR\_MIN; // CHAR\_MIN = -128 10000000 присваиваем это значение переменным uc и c  
 printf("CHAR\_MIN : c=%d uc=%d\n", c, uc); //выводим значения переменных uc и с  
 c = uc = UCHAR\_MAX; //UCHAR\_MAX = 255 11111111 присваиваем значение OxFF переменным uc и c  
 printf("UCHAR\_MAX : c=%d uc=%d\n", c, uc); //выводим значения переменных uc и с  
 c = c + 1; uc = uc + 1; //11111111 + = 00000000 прибавляем единицу к переменным uc и c  
 printf("UCHAR\_MAX+1 : c=%d uc=%d\n", c, uc); //выводим значения переменных uc и с  
 uc = c = -5; // 11111011 присваиваем значение OxFB переменным uc и c  
 printf("-5 : c=%d uc=%d\n", c, uc); //выводим значения переменных uc и с  
 c = -5; uc = 5; // присваиваем значение -5 переменной c, переменной uc 5  
 printf("char and unsigned char -5>5 : %d\n\n", c>uc); //выводим значение выражения  
 return 0;  
}

Результаты работы программы:

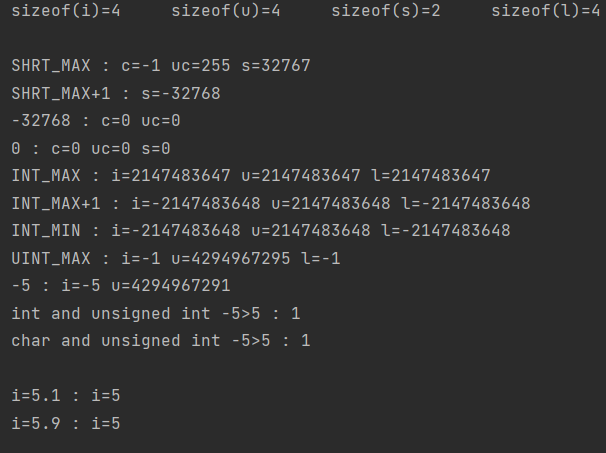


**Задание 7.**

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;  
 printf("sizeof(i)=%d\tsizeof(u)=%d\tsizeof(s)=%d\tsizeof(l)=%d\n\n",  
 sizeof(i), sizeof(u), sizeof(s), sizeof(l)); /\* вывод количества занимаемой памяти в байтах переменными \*/  
 c = s = SHRT\_MAX; /\* присваиваем переменной s значение 0x7fff, далее переменной c присваиваем значение переменной s, в переменной с происходит переполнение\*/  
 uc = s; /\* присваиваем значение переменной s переменной uc \*/  
 printf("SHRT\_MAX : c=%d uc=%d s=%d\n", c, uc, s);  
 s = s + 1; /\* прибавляем к переменной s единицу происходит переполнение \*/  
 printf("SHRT\_MAX+1 : s=%d\n", s);  
 c = s; uc = s; /\* присваиваем значение переменной s переменным c и uc \*/  
 printf("%d : c=%d uc=%d\n", SHRT\_MIN, c, uc);  
 s = 0; c = s; uc = s; /\* присваиваем переменной s значение 0, далее присваиваем переменным с и uc значение переменной s \*/  
 printf("0 : c=%d uc=%d s=%d\n", c, uc, s);  
 i = INT\_MAX; /\* присваиваем переменной значение 0x7fffffff \*/  
 l = i; u = i; /\* присваиваем переменным l и u значение переменной i \*/  
 printf("INT\_MAX : i=%d u=%u l=%ld\n", i, u, l); /\* вывод значений переменных i, u и l \*/  
 i = i + 1; l = l + 1; u = u + 1; /\* к значениям переменных i, l и u прибавляем 1 произойдет переполнение \*/  
 printf("INT\_MAX+1 : i=%d u=%u l=%ld\n", i, u, l);  
 i = INT\_MIN; /\* присваиваем переменной i значение ox800000000 \*/  
 l = i; u = i; /\* присваиваем переменным l и u значение переменной i\*/  
 printf("INT\_MIN : i=%d u=%u l=%ld\n", i, u, l);  
 u = UINT\_MAX; /\* присваиваем переменной u значение OxFFFFFFFF \*/  
 i = u; l = u; /\* присваиваем переменным l и i значение переменной u, в переменных l и i происходит переполнение \*/  
 printf("UINT\_MAX : i=%d u=%u l=%ld\n", i, u, l);  
 u = i = -5; /\* переменной i присваиваем значение -5, переменной u присваиваем значение переменной i, в переменной u другое значение из-за беззнакового типа \*/  
 printf("-5 : i=%d u=%u\n", i, u);  
 i = -5; u = 5; /\* переменной i присваивается значение -5, переменной u присваивается значение 5, i > u так как signed int повышается до unsigned int \*/  
 printf("int and unsigned int -5>5 : %d\n", i > u);  
 c = -5; u = 5; /\* переменной c присваивается значение -5, переменной u присваивается значение 5, c > u так как signed char повышается до unsigned int \*/  
 printf("char and unsigned int -5>5 : %d\n\n", c > u);  
 i = 5.1; /\* переменной i присваивается значение 5, так как дробная часть отбрасывается \*/  
 printf("i=5.1 : i=%d\n", i);  
 i = 5.9; /\* переменной i присваивается значение 5, так как дробная часть отбрасывается \*/  
 printf("i=5.9 : i=%d\n", i);  
 return 0;  
}

Результаты работы программы:

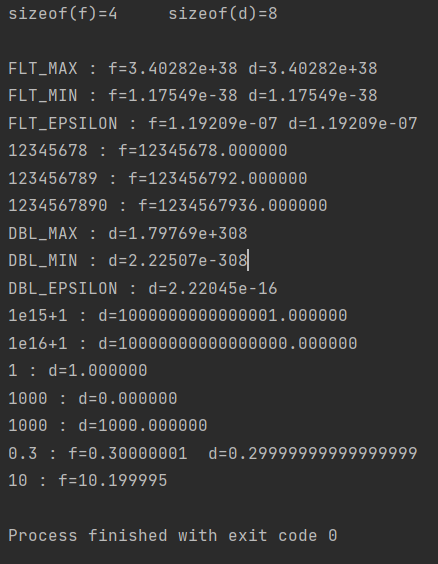
**

**Задание 8.**

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

#include <stdlib.h>  
#include <stdio.h>  
#include <float.h> /\* комментарии \*/  
  
int main()  
{  
 float f;  
 double d;  
 printf("sizeof(f)=%d\tsizeof(d)=%d\n\n", sizeof(f), sizeof(d)); /\* вывод количества занимаемой памяти в байтах переменными \*/  
 d = f = FLT\_MAX; /\* переменной f присваивается значение FLT\_MAX, переменной d присваивается значение переменной f \*/  
 printf("FLT\_MAX : f=%g d=%g\n", f, d);  
 d = f = FLT\_MIN; /\* переменной f присваивается значение FLT\_MIN, переменной d присваивается значение переменной f \*/  
 printf("FLT\_MIN : f=%g d=%g\n", f, d);  
 d = f = FLT\_EPSILON; /\* переменной f присваивается значение FLT\_EPSILON, переменной d присваивается значение переменной f \*/  
 printf("FLT\_EPSILON : f=%g d=%g\n", f, d);  
 f = 12345678; /\* значение 12345678 присваивается переменной f преобразовываясь в float \*/  
 printf("12345678 : f=%f\n", f);  
 f = 123456789; /\* значение 123456789 занимает 7 байт, происходит переполнение, так как по стандарту IEEE 754 мантисса занимает 23 бита \*/  
 printf("123456789 : f=%f\n", f);  
 f = 1234567890; /\* значение 123456789 занимает 8 байт, происходит переполнение, так как по стандарту IEEE 754 мантисса занимает 23 бита \*/  
 printf("1234567890 : f=%f\n", f);  
 d = DBL\_MAX; /\* переменной d присваивается значение DBL\_MAX \*/  
 printf("DBL\_MAX : d=%g\n", d);  
 d = DBL\_MIN; /\* переменной d присваивается значение DBL\_MIN \*/  
 printf("DBL\_MIN : d=%g\n", d);  
 d = DBL\_EPSILON; /\* переменной d присваивается значение DBL\_EPSILON \*/  
 printf("DBL\_EPSILON : d=%g\n", d);  
 d = 1e15 + 1; /\* переполнение не происходит число занимает 6 байт, по стандарту IEEE 754 мантисса занимает 52 бита \*/  
 printf("1e15+1 : d=%lf\n", d);  
 d = 1e16 + 1; /\* происходит переполнение, так как число занимает 7 байт, по стандарту IEEE 754 мантисса занимает 52 бита \*/  
 printf("1e16+1 : d=%lf\n", d);  
 d = 10000 \* 100000 + 1 - 4 \* 250000000; /\* комментарии \*/  
 printf("1 : d=%lf\n", d);  
 d = 1e20 \* 1e20 + 1000 - 1e22 \* 1e18; /\* комментарии \*/  
 printf("1000 : d=%lf\n", d);  
 d = 1e20 \* 1e20 - 1e22 \* 1e18 + 1000; /\* комментарии \*/  
 printf("1000 : d=%lf\n", d);  
 f = d = 0.3; /\* значение имеет бесконечную дробь в двоичном представлении, по стандарту IEEE 754 значение округляется \*/  
 printf("0.3 : f=%.8f d=%.17f\n", f, d);  
 f = 0;  
 while (f < 10) /\* значение имеет бесконечную дробь в двоичном представлении, по стандарту IEEE 754 значение округляется, при суммировании появляется погрешность \*/  
 f += 0.2;  
 printf("10 : f=%f\n", f);  
 return 0;  
}

Результаты работы программы:

**

**Задание 9.**

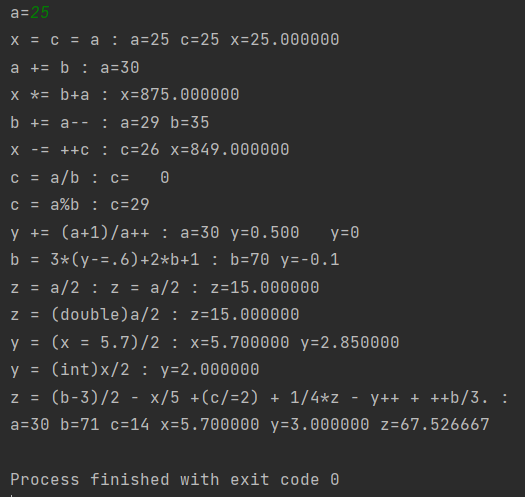
Проверить порядок выполнения операций в каждом выражении, содержащем несколько операций присваивания, разделив каждый оператор-выражение на несколько операторов, выполняемых последовательно.

Текст измененной программы:

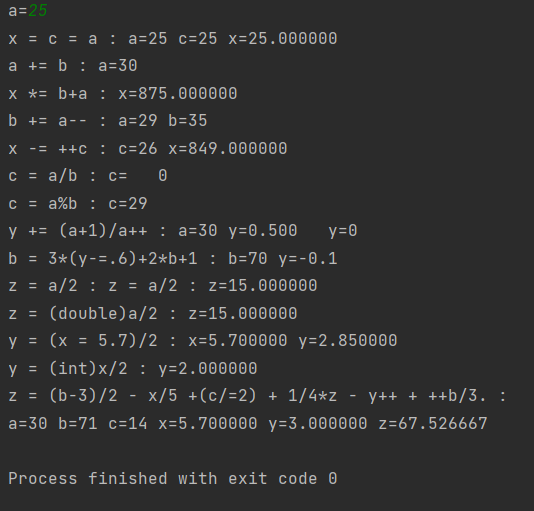
#include <stdio.h>  
#include <stdlib.h>  
  
int main() {  
 int a;  
 int b = 5;  
 int c;  
 double x;  
 double y = -.5;  
 double z;  
 printf("a=");  
 scanf("%d", &a);  
 c = a;  
 x = c;  
 printf("x = c = a : a=%d c=%d x=%f\n", a, c, x);  
 a = a + b;  
 printf("a += b : a=%d\n", a);  
 x = 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;  
 double t = 3 \* y;  
 b = t + 2 \* b;  
 b = 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);  
 z = (b-3)/2;  
 c = c/2;  
 z+= - x/5 + c + 1/4\*z;  
 ++b;  
 z += - y + b/3.;  
 y++;  
 printf("z = (b-3)/2 - x/5 +(c/=2) + 1/4\*z - y++ + ++b/3. :\n"  
 "a=%d b=%d c=%d x=%lf y=%lf z=%lf\n", a, b, c, x, y, z);  
 return 0;  
}

Результаты работы программ:

до изменения

**

после изменения

**

**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** |
| d = 5, b = 1.1 | a = 2.46, b = 0.1, c = 2, d = 27 | a = 2.46, b = 0.1, c = 2, d = 27 |
| d = 9, b = 7.3 | a = 6.18, b = 6.3, c = 2, d = 57 | a = 6.18, b = 6.3, c = 2, d = 57 |
| d = 13, b = 3.1 | a = 3.66, b = 2.1, c = 2, d = 93 | a = 3.66, b = 2.1, c = 2, d = 93 |

Текст программы:

#include <stdio.h>  
  
int main(){  
 int c;  
 int d;  
 double a;  
 double b;  
 c = a = 5;  
  
 scanf("%d %lf", &d, &b);  
 a +=b-2;  
  
 printf("%lf\t%lf\t%d\t%d\n",a,b,c,d);  
 d +=++c-a;  
  
 printf("%lf\t%lf\t%d\t%d\n",a,b,c,d);  
 a \*=c--;  
  
 printf("%lf\t%lf\t%d\t%d\n",a,b,c,d);  
  
 d \*= (c/=2) + --b + (a/=10);  
 printf("%lf\t%lf\t%d\t%d\n",a,b,c,d);  
 return 0;  
}