### Лабораторная работа №2. Арифметические операции и математические функции языка C

## 

### 1. Цель работы

Целью лабораторной работы является знакомство с арифметическими операциями и получение практических навыков в программировании алгеброичных выражений и использовании математических функций библиотеки языка C.

### 2. Темы для предварительной проработки

* арифметические операции
* порядок выполнения операций
* стандартные математические функции

*2.1 Операции языка СИ*

*Арифметические*

|  |  |
| --- | --- |
| - вычитание и унарный минус;  + сложение;  \* умножение;  / деление;  % деление по модулю;  ++ увеличение на единицу (х++ -> х=х+1);  -- уменьшение на единицу (х-- -> х=х-1);  x++ постфиксная форма;  ++х префиксная форма. | Пример:  int a;  int k=3, s=10;  a=11/3;  printf("\na=%d",a);  printf("\nk=%d\ts=%d",++k,s++);  printf("\nk=%d\ts=%d",k,s); |

*2.2 Операция присваивания*

|  |  |  |
| --- | --- | --- |
| x=2\*x;  a=b=c=x\*y; | m-=20;  m+=20;  m\*=20;  m/=20;  m%=20; | m=m-20;  m=m+20;  m=m\*20;  m=m/20;  m=m%20; |

*2.3 Операция запятая*

|  |  |
| --- | --- |
| Пример1:  int k;  printf("\n%d",(k=3,2\*k));  ); | Пример2:  char k[]="10";  printf("\n%d",sizeof(char));  printf("\n%d",sizeof(k)); |

*2.4 Операция sizeof*

sizeof(тип); sizeof(выражение)

*2.5. Порядок выполнения операций*

|  |  |  |
| --- | --- | --- |
| Приоритет | Операции | Направление выполнения |
| 1 | () (вызов функции), [], ->, "." | >>> |
| 2 | !, ~, +, - (унарные), ++, --, \*, (тип), sizeof, (new,delete - Си++) | <<< |
| 3 | .\* , ->\* - Си++ | >>> |
| 4 | \*, /, % (бинарные) | >>> |
| 5 | +, - (бинарные) | >>> |
| 6 | <<, >> | >>> |
| 7 | <, <=, =>, > | >>> |
| 8 | ==, != | >>> |
| 9 | & (поразрядное) | >>> |
| 10 | ^ | >>> |
| 11 | | (поразрядное) | >>> |
| 12 | && (логическое) | >>> |
| 13 | || (логическое) | >>> |
| 14 | ?: (условная) | <<< |
| 15 | =, +=, -=, \*=, /=, %=, &=, ^=, |=, <<=,>>= | <<< |

*2.6. Приведение типов*

|  |  |
| --- | --- |
| Пример *явного преобразования*:  double d=(double) 5;  int I = int (d); | Пример *неявного преобразования:*  long k=32768;  int a=k;  double d=0.9999;  long n=d;  k=32768  a=-32768  d=9.999000e-01  n=0 |

*2.7. Арифметические преобразования типов, гарантирующие сохранение значимости*



*2.8. Стандартные математические функции* (находятся в файле *math.h*.)

abs, acos, acosl, asin, asinl, atan, atanl, atan2, atan2l, atof, \_atoldcabs, cabsl, ceil, ceill, cos, cosl, cosh, coshl, exp, expl, fabs, fabslfloor, floorl, fmod, fmodl, frexp, frexpl, hypot, hypotl, labs, ldexp, ldexpl, log, logl, log10, log101, matherr, \_matherrl, modf,,modfl, poly, polyl, pow, powl, pow10, pow10l, sin, sinl, sinh, sinhl, sqrt, sqrtl, tan, tanl, tanh, tanhl

### 3. Задания для выполнения

Составьте программу, которая подсчитывает и выводит значение ***t1*** и ***t2*** по формулам, которые приведены в Вашем варианте индивидуального задания. Определите области допустимых значений параметров формул и задайте произвольные значения из этих областей. Параметры, которые имеют имена: ***n*** и ***m*** - целые, остальные параметры - с плавающей точкой. Значения параметров с именами ***x*** и ***y*** должны вводиться с клавиатуры, значения остальных - задаваться как начальные значения при объявлении соответствующих переменных. Допускается (и даже желательно) упростить / разложить формулы для того, чтобы обеспечить минимизацию объема вычислений.

### 4. Варианты индивидуальных заданий (см. приложение)

#### 5. Метод решения.

*5.1. Некоторые замечания.*

1. Выражение ***ax*** встречается один раз в первой формуле и дважды - во второй. Следовательно, можно один раз произвести умножение ***ax***, а потом использовать этот результат.
2. Во второй формуле дважды встречается умножение квадратного корня на тангенс - это вычисление можно так же сделать один раз.
3. Выражение ***c2-b2*** можно разложить на ***(c+b)(c-b)***. До разложения в выражении было две операции умножения (возведение в степень 2) и одна - сложения. После разложения - два сложения и одно умножение, что выгоднее для вычислений.
4. Аргумент функции, которую вычисляет логарифм, не может быть 0 или меньше. Отсюда вытекают требования к значениям: ***a · x + b > 0; y· x + d > 0***
5. Аргумент функции извлечения квадратного корня не может быть меньше 0, отсюда: ***c2 - b2 >= 0***
6. В знаменателе выражения не может быть 0, отсюда: l03_a1 кроме того:

#### 6. Выводы

При выполнении лабораторной работы изучены следующие темы: программирование алгебраических выражений с использованием арифметических операций и стандартных математических функций.

**7.Упражнения для самоконтроля**

1. Заполнить пробелы в следующих утверждениях:

1. Все программы можно писать в терминах трех типов управляю­щих структур: \_\_\_\_\_, \_\_\_\_\_\_ и \_\_\_\_\_\_.
2. Структура выбора \_\_\_\_\_\_ используется для выполнения одного действия, если условие истинно, и другого, если условие ложно.
3. Повторение ряда операторов заданное число раз называется \_\_\_\_\_\_ повторением.
4. Когда заранее не известно, сколько раз должна быть повторена группа операторов, можно использовать \_\_\_\_\_ для оконча­ния повторения.
5. Укажите неправильный идентификатор:
   1. AB\_D1
   2. 1xd
   3. z1d8\_14f3
6. Объявление переменой
   1. необходимо сделать до того, как использовать эту переменную
   2. можно сделать в любой момент до завершения работы
   3. можно сделать для лучшего понимания программы, но можно и опустить
7. Выберете правильное объявление константы пи:
   1. const float pi =3.14
   2. float pi = (const)3.14
   3. const float pi; pi =3.14
8. Отметьте фрагменты кода, которые можно назвать выражениями:
   1. f + r\*12-14
   2. if (x>0) y++
   3. x = y = 13
9. Чему равен результат вычисления выражения

x + 3 \* b + x

при x = 12 и b = 8?

* 1. 132
  2. 48
  3. 300

1. Каково будет значения переменной k после выполнения оператора k=k++, если до его выполнения k равнялось 6?
   1. 7
   2. 8
   3. 6

**Приложение**

|  |  |
| --- | --- |
| **Вариант 1** | **Вариант 2** |
| |  | | --- | | l03_e1 | | l03_e2 | | |  | | --- | | l03_e3 | | l03_e4 | |
| **Вариант 3** | **Вариант 4** |
| |  | | --- | | l03_e5 | | l03_e6 | | |  | | --- | | l03_e7 | | l03_e8 | |
| **Вариант 5** | **Вариант 6** |
| |  | | --- | | l03_e9 | | l03_e10 | | |  | | --- | | l03_e11 | | l03_e12 | |
| **Вариант 7** | **Вариант 8** |
| |  | | --- | | l03_e13 | | l03_e14 | | |  | | --- | | l03_e15 | | l03_e16 | |
| **Вариант 9** | **Вариант 10** |
| |  | | --- | | l03_e17 | | l03_e18 | | |  | | --- | | l03_e19 | | l03_e20 | |
| **Вариант 11** | **Вариант 12** |
| |  | | --- | | l03_e21 | | l03_e22 | | |  | | --- | | l03_e23 | | l03_e24 | |
| **Вариант 13** | **Вариант 14** |
| |  | | --- | | l03_e25 | | l03_e26 | | |  | | --- | | l03_e27 | | l03_e28 | |
| **Вариант 15** | **Вариант 16** |
| |  | | --- | | l03_e29 | | l03_e30 | | |  | | --- | | l03_e31 | | l03_e32 | |
| **Вариант 17** | **Вариант 18** |
| |  | | --- | | l03_e33 | | l03_e34 | | |  | | --- | | l03_e35 | | l03_e36 | |
| **Вариант 19** | **Вариант 20** |
| |  | | --- | | l03_e37 | | l03_e38 | | |  | | --- | | l03_e39 | | l03_e40 | |