# Линейные алгоритмы

# Выражение

*Выражение* – последовательность операторов, операндов и знаков пунктуации, воспринимаемую компилятором как руководство к определенному действию над данными.
*Инструкции* – выражения, завершающиеся точкой с запятой (;)

Пример:

```
int a, b(1);
a = b - 5;      // верно, но
//a = b ++ b;  // синтаксическая ошибка, следует
        // писать так:
a = b++; b;
        // или так:
a = b; ++b;
```

# Пустой оператор

Точка с запятой (;) является самостоятельным пустым оператором.

```
int a, b;
cin >> b;
a = b++;
; // ничего не значащая инструкция
;; // две пустых инструкции
if (a < 10)
   ; // оператор ";" составляет тело условного оператора
b--; // действие выполняется вне зависимости от
   // истинности условия "a < 10"
```

# Арифметические операции

| | |
|---|---|
| Сложение | + |
| Вычитание | - |
| Умножение | * |
| Деление | / |
| Остаток от целочисленного деления | % |

*оператор / ведет себя как целочисленное деление для двух целочисленных операндов

**оператор % применяется только к целочисленным операндам

# Примеры использования арифметических операций

```
int a = 5 / 3;                          // a = 1
int b = 5 % 3;                          // b = 2
int c = 5 / 2;                          // c = 2
int d = 5 % 2;                          // d = 1
double e = 5 / 2.0;                        // e = 2.5
double g = a / b;                       // g = 0.0
double f = 5. / b;                         // f = 2.5
int h = 5. / b;                            // h = 2
double i = (double) a / b;                    // i = 0.5
double j = a / static_cast<double>(b);     // j = 0.5
```

# Операции инкремента и декремента

префиксные (сначала операция применяется к операнду, затем его значение используется в выражении):

++a,  --a

постфиксные (сначала значение операнда используется в выражении, затем к операнду применяется операция):

a++,  a--

Пример:

```
int a = 2, b = 4;
int c = b * a++;  // c = 8,  a = 3;
int d = b * ++a;  // d = 16, a = 4;
```

# Модификации оператора присваивания

```
+=    a += b;    a = a + b;

-=    a -= b;    a = a - b;

*=    a *= b;    a = a * b;

/=    a /= b;    a = a / b;

%=    a %= b;    a = a % b;
```

# Операторы сравнения (отношения)

- возвращают результат типа bool

```
<       bool x = a < b;
<=      bool x = a <= b;
>       bool x = a > b;
>=      bool x = a >= b;
==      bool x = a == b;
!=      bool x = a != b;
```

# Приоритет операций

| Приоритет | Оператор | Описание | Ассоциативность |
|---|---|---|---|
| 2 | a++ a-- | Суффиксный/постфиксный инкремент и декремент | Слева направо |
| | a() | Вызов функции | |
| 3 | ++a --a | Префиксный инкремент и декремент | Справа налево |
| | +a -a | Унарные плюс и минус | |
| | (тип) | Приведение типов в стиле C | |
| 5 | a*b a/b a%b | Умножение, деление и остаток от деления | Слева направо |
| 6 | a+b a-b | Сложение и вычитание | |
| 9 | < <= | Операторы сравнения < и ≤ соответственно | |
| | > >= | Операторы сравнения > и ≥ соответственно | |
| 10 | == != | Операторы равенства = и ≠ соответственно | |
| 16 | = | Прямое присваивание (предоставляется по умолчанию для классов C++) | Справа налево |
| | += -= | Составное присваивание с сложением и вычитанием | |
| | *= /= %= | Составное присваивание с умножением, делением и остатком от деления | |
| 17 | , | Запятая | Слева направо |

*полная таблица приоритетов https://ru.cppreference.com/w/cpp/language/operator_precedence

# Математические функции

язык C++
заголовочный файл cmath:
double abs (double) - модуль числа
double pow (double, double) - возведения числа в степень

язык C
заголовочный файл math.h, используются те же функции, но:
int abs (int) – модуль числа типа int
double fabs (double) - модуль числа типа double
long labs (long) - модуль числа типа long
float fabsf (float) - модуль числа типа float
long double fabsl (long double) - модуль числа типа long double

# Математические функции

язык C++
double log (double) – натуральный логарифм
double log10 (double) – десятичный логарифм
log2
double exp(double) – возведение числа Эйлера *e* в степень
double sqrt (double) – квадратный корень
double cbrt - (double) – кубический корень

язык C
float logf(float) - натуральный логарифм типа float
long double logl(long double) - натуральный логарифм типа long double
float log10f (float) - десятичный логарифм типа float
long double log10l(long double) - десятичный логарифм типа long double
long double expl(long double) - возведение числа *e* в степень для long double
float sqrtf(float) – квадратный корень для типа float
long double sqrtl(long double) - квадратный корень для типа long double

# Прочие функции

язык C++
double hypot (double, double) – гипотенуза по катетам
double min (double, double) – наименьшее из двух
double max (double, double) – наибольшее из двух
void swap (double, double) – обмен двух значений
int div (int, int) – целочисленное деление

# Подключение заголовочных файлов

```
#include <имя_заголовочного_файла>
```

```
#include "имя_заголовочного_файла"
```

здесь:

- <имя_заголовочного_файла> - файл находится в папках, о которых знает компилятор (обычно папка "include", можно изменить в настройках IDE);
- "имя_заголовочного_файла" – файл находится в локальной папке (обычно в папке с текущим проектом, если задано только имя файла).

В языке C++ допустимо подключать заголовочные файлы – обертки над заголовками C, для этого к имени заголовка C дописывается префикс "c", а расширение файла не указывается. Например, вместо

```
#include <math.h>
```

используем

```
#include <cmath>
```

# Функции fmod и modf

- double fmod(double, double) – вычисление остатка от деления при работе с дробными числами;
- double modf(double, double*) – разделение дробного числа на целую и дробную части.

# Пример работы с функциями fmod и modf

Определить остаток от деления дробного числа *a* на дробное число *b*.
Вычислить целую и дробную часть дробного числа *c.*

```
double a(3.1), b(3);
double f = fmod(a, b); // f = 0.1
double c, fracPart, intPart;
c = 3.5;
fracPart = modf(c, &intPart);
// после исполнения этой строки в fracPart хранится
// дробная часть c, в intPart – целая часть
// fracPart = 0.5
// intPart = 3.0
```

# Основные математические константы

- M_PI – значение числа $\pi$
- M_PI_2 – значение $\frac{\pi}{2}$
- M_PI_4 – значение $\frac{\pi}{4}$
- M_E – значение числа Эйлера $e$
- M_LOG2E – значение $\log_2 e$
- M_LOG10E – значение $\log_{10} e$
- M_LN2 – значение $\ln 2$
- M_LN10 – значение $\ln 10$
- M_SQRT2 – значение $\sqrt{2}$

*для некоторых компиляторов необходимо дополнительно использовать директиву
#define _USE_MATH_DEFINES

# Тригонометрические функции (значение задается и вычисляется в радианах)

язык C++

```
double cos(double);
double cosh(double);
double sin(double);
double sinh(double);
double tan(double);
double tanh(double);
double acos(double);
double asin(double);
double atan(double);
```

язык C

```
float cosf(float);
float coshf(float);
float sinf(float);
float sinhf(float);
float tanf(float);
float tanhf(float);
float acosf(float);
float asinf(float);
float atanf(float);
```

# Пример разбора математического выражения

Вычислить: $rez = \sqrt{\left|x^3 + \ln y - \lg z\right|}$

```
pow(x, 3)
```
$x^3$

```
pow(x, 3) + log(y)
```
$x^3 + \ln y$

```
pow(x, 3) + log(y) - log10(z)
```
$x^3 + \ln y - \lg z$

```
abs(pow(x, 3) + log(y) - log10(z))
```
$\left|x^3 + \ln y - \lg z\right|$

```
sqrt(abs(pow(x, 3) + log(y) - log10(z)))
```
$\sqrt{\left|x^3 + \ln y - \lg z\right|}$

```
double rez = sqrt(abs(pow(x, 3) + log(y) - log10(z)));
```

# Тригонометрические функции

double atan2(double x, double y);

float atan2f(float, float);

Вычислить: $\dfrac{\sin x + \cos y}{arctg\dfrac{y}{x}}$

```
double x, y;
…
double rez = (sin(x) + cos(y)) / atan2(y, x);
// или так
double rez = (sin(x) + cos(y)) / atan(y / x);
```

# Функции округления

double floor (double) – округление "вниз"

double ceil (double) – округление "вверх"

double round(double) – округление по правилам округления

Пример:

```
double a = 7.765432;
double b = floor(a); // b = 7.0
double c = ceil(a);  // c = 8.0
double d = round(a);  // d = 8.0
```

# Функции ГСЧ (cstdlib)

int rand() - возвращает случайное целое число в диапазоне от 0 до RAND_MAX
void srand (unsigned seed) - устанавливает новое зерно ГСЧ.

Например:

```
// целое число от 0 до RAND_MAX
int rnd1 = rand();

// целое число от 0 до 9
int rnd2 = rand() % 10;

// дробное число от 0 до 10
// с одним знаком после запятой, 10 не включается
float rnd3 = rand() % 100 / 10.0;
```

# Использование srand

```
#include <ctime>
#include <cstdlib>
#include <iostream>
using namespace std;

int main(){
srand(time(NULL));
cout << rand();
}
```

# Функция sizeof()

Функция `sizeof()` позволяет определить размер типа (или программного объекта) данных в байтах:

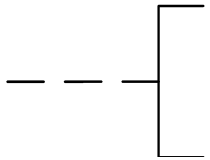
```
sizeof (char);    // 1
sizeof (float);     // 4
short var;
sizeof(var);   // 2
```

# Линейные алгоритмы в виде схем алгоритмов

Начало – конец

Ввод-вывод

Процесс



Комментарий

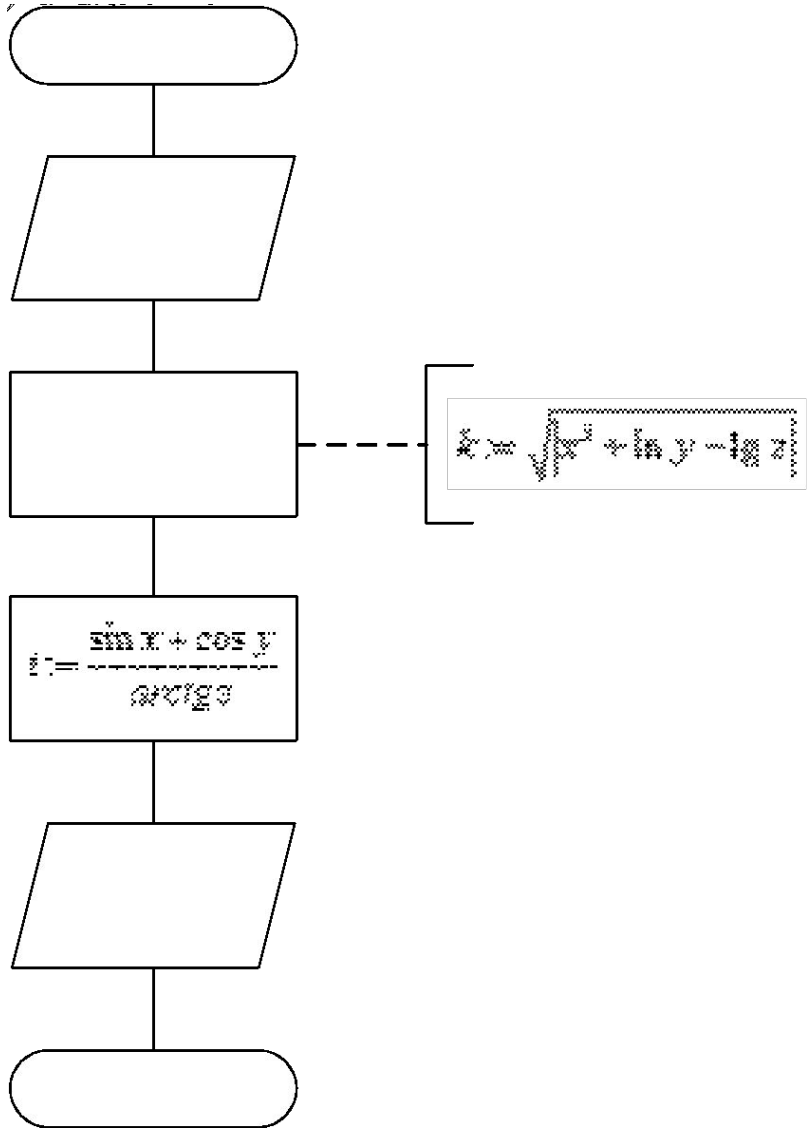
# Линейные алгоритмы в виде схем (пример программы)

Даны действительные x, y, z.

Вычислить:

1. $k=\sqrt{\left|x^3+\ln y-\lg z\right|}$

2. $t=\dfrac{\sin x+\cos y}{arctg\, z}$



# Пример решения задачи 1

Вычислить выражение $tg^2 x^3 - \frac{\ln y}{x+1}$, где *x* и *y* вводятся с клавиатуры.

```
#include <cmath>
#include <iostream>
using namespace std;

int main(){
float x, y, z;
cin >> x >> y >> z;
float z = pow(tan(pow(x, 3)), 2) - log(y) / (x + 1);
cout << z;
}
```

# Пример решения задачи 2

Известны стороны *a* и *b* прямоугольника. Найти площадь прямоугольника.

```
#include <iostream>
using namespace std;

int main(){
float a, b, s;
cin >> a >> b;
float s = a * b;
cout << s; // cout << a * b;
}
```

# Пример решения задачи 3

Известен радиус окружности *r*. Найти округленное в большую сторону значение площади окружности.

```
#include <cmath>
#include <iostream>
using namespace std;

int main(){
float r;
cin >> r;
double s = M_PI * pow(r, 2);
cout << ceil(s);
}
```

# Пример решения задачи 4

Дано трехзначное число *n*. Найти сумму цифр данного числа

```
#include <iostream>
using namespace std;

int main(){
unsigned short n;
cin >> n;
unsigned short n1 = n / 100;
unsigned short n2 = n / 10 % 10;
unsigned short n3 = n % 10;
cout << n1 + n2 + n3;
}
```

# Пример решения задачи 5

Дано трехзначное число *n*. Первую и третью цифру числа поменять местами.

Значение трехзначного числа, в котором n1 – первая цифра, n2 – вторая, n3 – третья, определяется следующим образом:

$$n1 \cdot 10^2 + n2 \cdot 10 + n3$$

Следовательно после отделения цифр числа необходимо составить новое число с другим порядком следования цифр, например:

…

```
cout << n3 * 100 + n2 * 10 + n1;
```

# Дополнительно

# Создание синонима типа (typedef)

```
typedef unsigned short us;
us i; // i – переменная типа unsigned short
```

# Целочисленные типы с фиксированным размеров (C++11)

```
int8_t
int16_t
int32_t
int64_t
```

…

* подробнее https://en.cppreference.com/w/cpp/types/integer

# Альтернатива const

```
#define someConst2 123;
```

Отличия от описания константы с помощью const:

- не действует область видимости;
- отсутствует тип;
- не воспринимается как объект в памяти.

# Суффиксы и префиксы в литералах

Можно уточнить тип литерала с помощью суффиксов *f* (float), *l* (long или long double), *u* (unsigned) и пр.

Например,

- `2E-2L` – long double - 0.02
- `3.24f` – float,
- `20000L` – long,
- `20LL` – long long.

Можно уточнить систему счисления с помощью префикса:

- `02` – 8-ричная,
- `0x2` – 16-ричная.

*целый тип по умолчанию - signed int, поэтому для long unsigned исп суффикс LU UL L U lu ul l u

**действительный тип по умолчанию - double, поэтому для float и long double  исп суф f F l L при записи с десятичной точкой

# Заголовочный файл float.h

Содержит информацию о значениях типов с плавающей точкой:

- FLT_MIN, FLT_MAX – минимальное и максимальное значение типа float;
- DBL_MIN, DBL_MAX – минимальное и максимальное значение типа double;
- LDBL_MIN, LDBL_MAX – минимальное и максимальное значение типа long double;

# Диапазоны действительных типов в разных средах разработки

```
#include <iostream>
#include <float.h>
using namespace std;
int main() {
cout << sizeof(long double) << endl
     << LDBL_MIN << endl
     << LDBL_MAX << endl;
cout << sizeof(double) << endl
     << DBL_MIN << endl
     << DBL_MAX << endl;
cout << sizeof(float) << endl
     << FLT_MIN << endl
     << FLT_MAX << endl;
}
```

Qt

```
16
3.3621e-4932
1.18973e+4932
8
2.22507e-308
1.79769e+308
4
1.17549e-038
3.40282e+038
```

Code::Blocks

```
12
3.3621e-4932
1.18973e+4932
8
2.22507e-308
1.79769e+308
4
1.17549e-038
3.40282e+038
```

Visual Studio

```
8
2.22507e-308
1.79769e+308
8
2.22507e-308
1.79769e+308
4
1.17549e-38
3.40282e+38
```

# Указание не закрывать консоль при завершении работы программы

- system("pause");
- cin >> …;
- cin.get(…);

# Использование кириллицы в окне консольного приложения

1. В настройках параметров консольного окна установить название шрифта - Lucida Console.
2. Для обеспечения вывода текста, содержащего кириллицу, добавить в main вызов одной из функций:

- setlocale(LC_ALL, "Russian") - в редакторе кода необходимо использовать кодировку UTF-8;
- SetConsoleOutputCP(1251) - необходимо подключить заголовочный файл Windows.h.

3. Для обеспечения ввода текста, содержащего кириллицу, добавить в main вызов одной из функций:

- system("chcp 1251");
- SetConsoleCP(1251) - необходимо подключить заголовочный файл Windows.h.

# Значения вне диапазона типа при вычислениях (переполнение)

```
signed int si = 1500000000;
unsigned int ui = 1500000000;
si = si * 2 / 3;                    // неверно
si = si / 3 * 2;                    // верно
ui = ui * 2 / 3;                    // верно
ui = (ui * 10) / 10;                    // неверно
ui = (ui / 10) * 10;                    // верно
ui = (static_cast<double>(ui) * 10) / 10;  // верно
```