ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Лекция № 6

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Программы бывают линейными, разветвленными, циклическими и сложными. Большинство программ являются сложными.

Любую программу можно разбить на линейные, разветвленные и циклические фрагменты.

Наиболее простыми являются линейные программы или линейные фрагменты сложных программ.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Линейные алгоритмы

Линейными называются программы, в которых операторы выполняются один за другим с первого до последнего, не повторяясь и не изменяя порядка их выполнения.

Линейные программы могут содержать операторы присваивания, математические функции, арифметические операции, действия, связанные с вводом-выводом, и другие операторы, не изменяющие порядка следования операторов в программе.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Основные математические функции языка С++

Функция	*Обозначение функции*	*Тип*		*Имя файла описания*
		возвращаемого значения функции	*аргумента*	
Абсолютное значение	abs(x) cabs(x) fabs(x)	int double Float	int double float	<stdlib.h> <math.h> <math.h>
Арккосинус	acos(x)	Double	double	<math.h>
Арксинус	asin(x)	Double	double	<math.h>
Арктангенс	atan(x)	Double	double	<math.h>
Косинус	cos(x)	Double	double	<math.h>
Синус	sin(x)	Double	double	<math.h>
Экспонента ex	exp(x)	double	double	<math.h>
Степенная функция (x, y)	pow(x,y)	double	double	<math.h>
Логарифм натуральный	log(x)	double	double	<math.h>
Логарифм десятичный	loglO(x)	double	double	<math.h>
Корень квадратный	sqrt(x)	duble	duble	<math.h>
Тангенс	tan(x)	duble	duble	<math.h>

Заголовочный файл, необходимый для работы этих функций:

<math.h> или <cmath.h>

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Пример 3.1. **Составить программу для вычисления объема V и площади поверхности S полого шара по заданным внешнему и внутреннему радиусам R и r, если известно, что $S = 4\pi(R^2 + r^2)$, $V = 4/3\pi(R^3 - r^3)$.**

```
#include <iostream>
using namespace std;
#include <cmath.h>
const double Pi = 3.1415926;  // опредепение числа π
int main()
{      double S,V, R, r;
cout << "Введите внешний радиус";
cin >> R;
cout << "Введите внутренний радиус ";
cin >> r;
S = 4Pi(R*R - r*r);
V = 4.0/3*Pi*(pow(R,3)-pow(r,3));
cout << "S = " S << endl;
cout << "V = " << V << endl;
return 0; }
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Пример 3.2. **Ввести координаты точек (xl, yl) и (x2, y2). Определить расстояние между этими точками.**

```
#include <iostream>
using namespace std;
#include <conio.h>
#include <cmath.h>
int main() {int x1,x2,y1,y2;
double dist;
cout << "Введите x1 ->";
cin >> x1;
cout << "Введите x2 ->";
cin >> x2;
cout << "Введите y1 ->";
cin >> y1;
cout << "Введите y2 ->";
Cin >> y2;
dist = sqrt(pow((x1-x2),2)+pow((y1-y2),2));
cout << "Расстояние равно " << dist << endl;
getch();          // подключается с помощью заголовочного файла conio.h, останавливает выполнение программы до нажатия любой клавиши return 0;
}
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Пример 3.3. **Поменять местами значения переменных x и y.**

```
#include <iostream>
using namespace std;
#include <conio.h>
#include <cmath.h>
int main()
{int x, y, wrk;      // wrk - рабочая переменная
cout << "Введите x и y ->";
cin >> x >> y);
cout << "x = " << x << " y = " << y << endl;
wrk = x;       // запомнить в wrk значение переменной x
x = y;         // поместить в x значение y
y = wrk;       // поместить в y значение x, хранящееся в wrk
cout << "x = " << x << " y = " << y << endl;
getch();
return 0; }
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Помимо потоковых операций ввода-вывода cin и cout часто используются функции, применяемые в языке С: **printf() и scanf()**, которые предназначены для реализации форматного вывода и ввода данных.

Функция printf() имеет следующий синтаксис:
printf ("управляющая_строка", [список_аргументов])

Список аргументов - это последовательность констант, переменных или выражений, значения которых выводятся на экран дисплея в соответствии с форматом управляющей строки.

Список аргументов в функции printf() может отсутствовать.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Управляющая строка содержит объекты трех типов:

- обычные символы, выводимые на экран без изменений;
- спецификации преобразования, каждая из которых вызывает вывод на экран значения очередного аргумента из последующего списка аргументов;
- управляющие символьные константы.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Каждая спецификация преобразования начинается с символа % и заканчивается символом преобразования. Между ними могут записываться:

- знак минуса (-), указывающий на то, что выводимый текст выравнивается по левому краю, по умолчанию выравнивание происходит по правому краю;
- строка цифр, задающая минимальный размер поля вывода;
- точка, являющаяся разделителем;
- строка цифр, задающая точность вывода;
- символ l, указывающий на то, что соответствующий аргумент имеет тип long.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Символ преобразования может быть следующим:

d - аргумент преобразуется в десятичное представление;
o - аргумент преобразуется в восьмеричное представление;
x - аргумент преобразуется в шестнадцатеричное представление;
c - значением аргумента является символ;
s - значением аргумента является строка символов;
e - значением аргумента является величина типа float или double в экспоненциальной форме записи;
f - значением аргумента является величина типа float или double в форме записи с десятичной точкой;
g - один из форматов f или e;
u - значением аргумента является целое беззнаковое число;
p - значением аргумента является указатель (адрес).

Таким образом, «управляющая_строка» определяет количество и тип аргументов

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Среди управляющих символов наиболее часто используются следующие:

\a - кратковременный звуковой сигнал;
\n - перевод строки;
\t - горизонтальная табуляция;
\b - возврат курсора на один шаг назад;
\r - возврат каретки.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Функция scanf описывается аналогично функции printf:
scanf ("управляющая_строка", [список_аргументов]);
но список аргументов здесь является обязательным и не может отсутствовать.

Аргументы функции scanf() должны быть указателями на соответствующие значения, для этого перед именем переменной записывается символ &. Управляющая строка содержит спецификации преобразования и используется для определения количества и типов аргументов, аналогично функции printf().

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Пример 3.4:

printf ("i=%d,\n j=%d, a=%6.2f.\n", i, j, a);

Если i=1234, j=127, a=86.531, то на экране увидим:

i =1234,
j =127, a = 86.53.

В управляющей строке допустимо использование символов заполнения, по умолчанию в качестве символа заполнения применяется пробел.

printf ("i=%00d,\n j=%d,a=%6.2f.\n",i,j,a);

Теперь значение i выглядит так:

i =001234,

scant ("%d %f %c %s", &i,&a,&ch,r);

Здесь r - строка символов, имя которой само является указателем, поэтому перед ней знак амперсанда (&) не ставится.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Если входные данные при вводе с клавиатуры разделяются разделителями, например запятыми, то и в управляющей строке спецификации преобразования должны быть разделены такими же разделителями. Перепишем пример 3.1 для случая использования операций printf() и scanf().

Пример 3.5:

```
#include <stdio.h>      // подключение заголовочных файлов
#include <math.h>
#define Pi 3.1415926 // определение символической константы с этого момента и на протяжении всей программы Pi определена как 3.1415926
void main() {double S,V;
float R,r;
printf("ВВедите через запятую внешний и внутренний радиусы R и г:");
scanf("%t ,%f", &R,&r);
printf("S=%g,V=%t\n\S=4 * Pi * (R * R-r * r), V=4/3 * Pi * (R * R * R-r * r * r));
}
```

Разветвленные алгоритмы

Разветвленные алгоритмы предусматривают выбор маршрута выполнения программы в зависимости от истинности или ложности некоторых условий. Это обеспечивается наличием в программе специальных операторов, которые иногда называют *конструкциями принятия решений*:

if

if - else

switch.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Синтаксис оператора **if** имеет вид:

if (выражение) оператор;

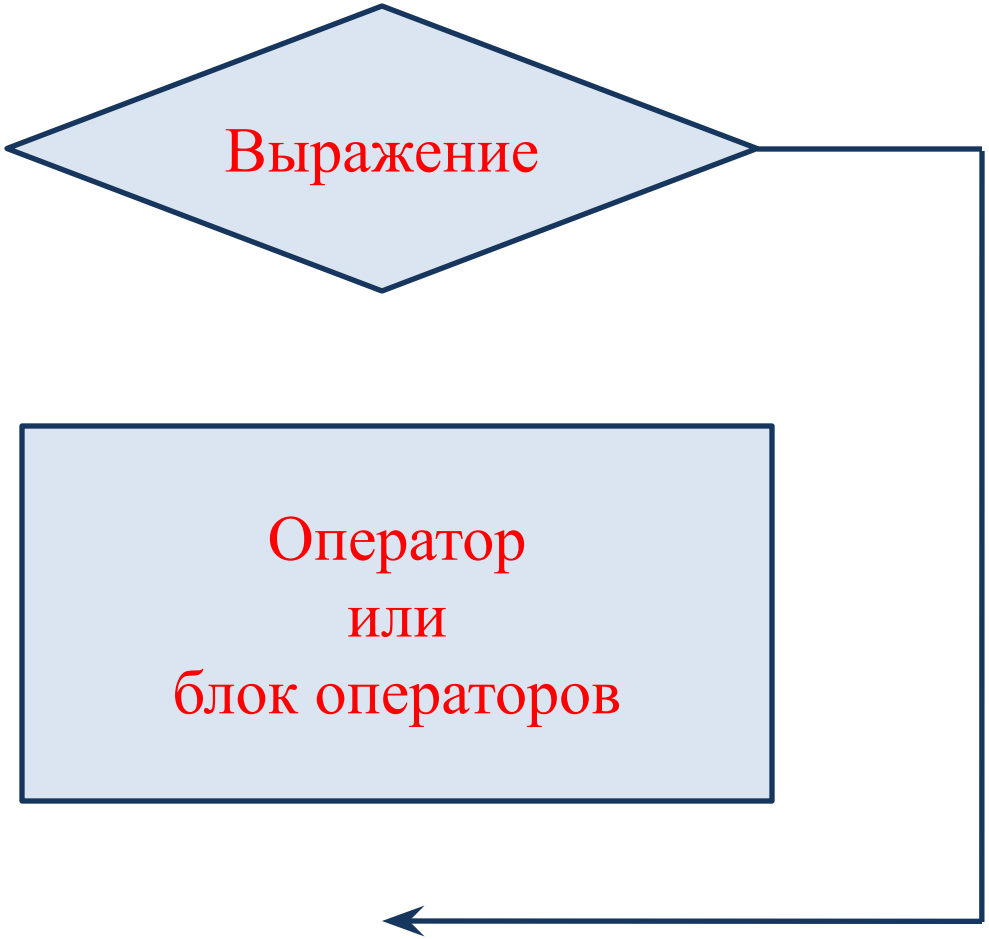
Если оператор, выполняемый при истинности условия выражения, единственный или если таких операторов несколько:

```
if (выражение) {оператор1;
                оператор2;
                …
                    операторN;
}
```

Здесь под выражением понимается любое логическое выражение или любое выражение, значение которого приводимо к целочисленному значению. Если его значение истинно, то оператор будет выполняться.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Схема алгоритма оператора if



ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

При необходимости сравнить выражение с некоторым значением следует использовать операции отношений в виде:
if **(выражение = = значение) оператор;**
или
if (выражение!=значение) оператор;
или
if (выражение > значение) оператор;
или
if (выражение < значение) оператор;

Нельзя писать **if (выражение =значение) оператор;**

Это одна из наиболее распространенных и труднообнаруживаемых ошибок.

Результатом такого использования операции присваивания «=» будет сравнение выражения со значением с последующим присвоением выражению значения, с которым оно сравнивалось.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Пример *3.6.* Использование оператора if для определения абсолютной величины введенного с клавиатуры целого значения.

```
#include <iostream>
using namespace std;
int main() {
int dig;
cout << "Введите число:";
cin >> dig;
if (dig < 0) dig = -dig;
cout << dig << endl;
return 0;
}
```

Выражение, служащее условием, заключается в круглые скобки.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

***Пример** 3.7.* По номеру y > 0 некоторого года определить c - номер его столетия. Учесть, что, например, XXI в. начинается с 2001 г., а не с 2000-го.

```
#include <iostream>
using namespace std;
int main() {
int dig;
cout << "Введите год:";
cin >> y;
c = y/100;
if (y%100!=0) c+=1;
cout << "Этот год принадлежит к веку " << c << endl;
return 0;
}
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Оператор if- else

Оператор if может иметь две ветви, одна из которых является альтернативной. Синтаксис оператора if - else имеет вид:

if (выражение) оператор1;
else оператор2;

если после if и else находится по одному оператору, или

if **(выражение)** {**оператор-!;**
оператор2;}
else {**оператор3;**
оператор4;},

если после if и else находится блок операторов, т. е. два или больше оператора, заключенные в фигурные скобки. Точка с запятой после фигурной скобки, закрывающей блок операторов, не ставится.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Схема алгоритма оператор if - else

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Вложенные ветвления

В качестве внутренних операторов оператора if могут использоваться любые операторы, в том числе и условные. Другими словами, в операторе if допустимо применение вложенных конструкций:

```
if (выражение1) оператор1;
else if (выражение2) оператор2;
else if (выражение3) оператор3;
else if (выражениеN) операторN;
else      // необязательная часть
     оператор_по_умолчанию;
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Вложенные ветвления

В качестве внутренних операторов оператора if могут использоваться любые операторы, в том числе и условные. Другими словами, в операторе if допустимо применение вложенных конструкций:

if **(x<=0){ y=e^x+x; cout<<“y=e^x+x”<<y<<endl; }**
else if **(x>0&&x<=7) {tg(3*x); cout<<“y=tg3*x”<<y<<endl;}**
else if **(x>7) (y=x-6; cout<<“y=x-6”<<y<<endl;}**

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

В подобных конструкциях часть else связывается с ближайшим предыдущим if в том же блоке, не имеющем части else. Последний оператор else, за которым следует **оператор_по_умолчанию;,** не является обязательным. Формально уровней вложенности операторов if может быть много, но реально при количестве таких вложенных конструкций, большем чем 4-5, программа становится трудноотлаживаемой.

Пример 3.8. :

```
#include <iostream>
using namespace std;
int main () {int value;
cout << "Input value from 1 to 10:";
cin >> value;
if (value >= 1)
if (value > 10) cout >> "Error: value >10" << endl;
    else cout << "Error: value <1“ << endl;
return 0; }
/* Поскольку else связывается с ближайшим if, эта программа будет работать неправильно */.
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

```
#include <iostream>
using namespace std;
int main ()
{int value;
cout << "Input value from 1 to 10: ";
cin >> value;
if (value >= 1)
{
    if (value > 10) cout << "Error: value >10" << endl;
}
else cout << "Error: value <1" << endl;
return 0;
}
/* Фигурные скобки скорректировали поведение программы.
Теперь все работает так, как было задумано */.
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Приведенный пример наглядно демонстрирует, что во избежание неоднозначного толкования программы следует пользоваться фигурными скобками, не полагаясь в сомнительных случаях на компилятор. При этом отступы, как и комментарии, носят чисто декоративный характер и компилятором игнорируются.

Вычислить d - max(a,b,c); - максимальное из трех введенных с клавиатуры чисел.

```
#include <iostream>
using namespace std;
int main() { int a, b, c, d;
cout << "a > ";
cin >> a;
cout >> "b > ";
cin >> a;
cout << "c > ";
cin >> a;
if (a>b && a>c) d = a;
   else if (b>c) d = b;
      else d = c;
cout << "max = " << d << endl;
return 0; }
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Условная операция

Условная операция является трехоперандной и имеет синтаксис:

переменная = выражение? значение1: значение2;

Такая запись является аналогом условного оператора

if (выражение) переменная = значение1;

else переменная = значение2;

Условный оператор и условное выражение в результате компиляции формируют практически идентичный код. Разница состоит в том, что в случае условного оператора обращение к переменной происходит дважды; следовательно, дважды вычисляется ее адрес, а в случае условной операции - лишь один раз. С другой стороны, с точки зрения понимания программы условный оператор намного лучше.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Пример 3.10:

```
if (test = = 'Y')
   TestValue = 100;
   else
     TestValue = 0;
```

что полностью эквивалентно:

```
TestValue = (test = = 'Y')?100:0;
```

Пример 3.11:

```
if (a>b) max = a;
     else max = b;
```

эквивалентно: `max = (a>b)?a:b;`

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Оператор множественного выбора

Если в программе необходимо выбрать один из многочисленных вариантов, то вместо вложенных конструкций if - else более целесообразно применять оператор-переключатель switch, иначе называемый *оператором множественного выбора.*

Его синтаксис:

switch **(выражение)**
{case **значение1: оператор1;**
break;
case **значение2: оператор2;**
break;
case **значение3: оператор3;**
break;
default: // необязательный компонент
оператор_по_несравнению; // если не было ни одного совпадения.
} //end switch **(выражение)**

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

- Фигурные скобки, ограничивающие тело оператора switch, являются обязательными. Здесь для выполнения выбирается тот вариант (группа операторов), значение которого совпадает со значением выражения.
- Оператор в каждом блоке выбора case может быть отдельным оператором или блоком операторов.
- Оператор break в каждом блоке выбора case осуществляет выход из оператора switch.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Пример 3.12. Ввести с клавиатуры символ. Если он является символом арифметической операции, то указать, какой именно, и привести соответствующий пример. Если не является, то выдать сообщение об этом.

```
#include <iostream>
using namespace std;
int main() { char sym;
int op1, op2, res;
cout << "Введите символ арифметической операции "
cin >> sym;
switch (sym) { case'+':   cout << "Сложение <<  endl;
                  cout << "1 слагаемое:";
        cin >> op1;
        cout << "2 слагаемое:";
        cin << op2;
        res = op1 + op2;
        cout << op1 << '+‘ << op2 << '=‘ << res << endl;
        break;
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

```
case '-':           cout << "Вычитание" << endl;
        cout << "Уменьшаемое:";
        cin << op1;
        cout << "Вычитаемое:";
        cin >> op2;
        res = op1 - op2;
        cout << op1 << '-' << op2 << '=' << res << endl;
        break;
case '*':     cout << "Умножение" << endl;
        cout << "Множимое:";
        cin >> op1;
        cout << "Множитель:";
        cin >> op2;
        res = op1 * op2;
        cout << op1 << '*‘ << op2 << '=‘ << res << endl;
        break;
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

```
  case '/':      cout << "Деление“ << endl;
            cout << "Делимое:";
            cin >> op1;
            cout << "Делитель:";
            cin >> op2;
            if (op2 != 0)
                 {res = op1 / op2;
                 cout << op1 <<  '/'  << op2 << '=‘ << res << endl;
                  } else cout << "Деление на 0 запрещено " << endl;
            break;
   default: cout << "Неарифметическая операция" << endl;
   }  // end switch (sym)
  return 0;
}
```

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Контрольные вопросы

- Что такое линейные и разветвленные программы?
- В чем заключается понятие форматного вывода данных?
- Как осуществляется ввод данных?
- Каковы формы условного оператора?
- Каким образом строится условное выражение?
- Какую структуру имеет оператор-переключатель?
- С какой целью используется оператор разрыва break?
- Что происходит, если забыть поставить break?
- Определить значение переменной w после выполнения следующих операторов:

```
w= 100; u = 30;
switch (u/7)
{ case 0: w = 0; break;
case 1: w = 1; break;
case 2: w = 2; break;
case 3: w = 3; break;
default: w = 7; }
```

10. Определить значение переменной m после выполнения следующих операторов:

```
т = 5;
if (x>0) {if (y>0) m = 10;} else m = 20;
```

а) при x = -5, y = 7;
б) при x = 2, y = -3;
в) при x = 9, y = 3.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Контрольные вопросы

- Что такое линейные и разветвленные программы?
- В чем заключается понятие форматного вывода данных?
- Как осуществляется ввод данных?
- Каковы формы условного оператора?
- Каким образом строится условное выражение?
- Какую структуру имеет оператор-переключатель?
- С какой целью используется оператор разрыва break?
- Что происходит, если забыть поставить break?
- Определить значение переменной w после выполнения следующих операторов:

```
w= 100; u = 30;
switch (u/7)
{ case 0: w = 0; break;
case 1: w = 1; break;
case 2: w = 2; break;
case 3: w = 3; break;
default: w = 7; }
```

10. Определить значение переменной m после выполнения следующих операторов:

```
т = 5;
if (x>0) {if (y>0) m = 10;} else m = 20;
```

а) при x = -5, y = 7;
б) при x = 2, y = -3;
в) при x = 9, y = 3.

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

Практические задания

1. **Вычислить S=A+B+C+D, если хотя бы одно из чисел A, B, C, D равно нулю, и P=A*B*O*D, если все числа отличны от нуля.**
2. **Составить программу, которая при вводе оценки в виде цифры выводит оценку в буквенном виде: 5 - отлично, 4 - хорошо, 3 - удовлетворительно, 2 – неудовлетворительно.**
3. **Составить программу, которая по введенному номеру месяца выводит его название и время года.**
4. **Составить программу, которая при вводе символа определяет, скобка ли это, и указывает, какая именно, например фигурная открывающая ({), квадратная закрывающая (]).**
5. **Составить программу, которая при вводе символа выводит либо текст «цифра», если введена цифра, либо текст «латинская буква», если введена латинская буква, либо текст «не цифра и не латинская буква» в остальных случаях.**
6. **Ввести с клавиатуры координаты точек (xl, yl) и (x2, y2) и определить расстояние между этими точками.**
7. **Поменять местами значения переменных x и y.**

ЛИНЕЙНЫЕ И РАЗВЕТВЛЕННЫЕ АЛГОРИТМЫ

8. **Значения переменных a, Ь, с поменять местами так, чтобы оказалось a>Ь>c.**
9. **Определить: y = max(min(a,b), min(c,d)).**
10. **Определить: y = min(a,b,c).**
11. **Переменной k присвоить номер четверти координатной плоскости, в которой находится точка с координатами, введенными с клавиатуры. Отдельно учесть случаи, когда точка попадает на одну из координатных осей или в начало координат.**
12. **Написать программу преобразования прописных латинских букв в строчные. При написании программы использовать условное выражение.**
13. **Написать программу решения квадратного уравнения с произвольными коэффициентами, которые вводятся с клавиатуры.**
14. **Определить d = max(a, b, c), если значения переменных a, b и c введены с клавиатуры.**