

# Управление

- *Выражения* – вычисление значений
- *Операторы* – последовательность изменений состояния памяти
- *Процедуры и функции* – подпрограммы, рекурсия
- *Обработка исключительных ситуаций* – восстановление после ошибок

# Выражения

Выражения строятся из

- имён переменных
- литеральных значений и имён констант
- применения операций
- разыменованя, взятия адреса, выборки компонент массивов и структур
- явного приведения типа и вычисления размера типа
- вызова функций и процедур
- группирования вычислений скобками
- условного и последовательного выражений

# Выражения

$+, -, *, /, \%, \&, |, ^, \ll, \gg$

$==, !=, <, >, <=, >=, \|\|, \&\&$

$=, +=, -=, *=, /=, \%=, \&=, |=, ^=, \ll=, \gg=$

$++, --, +, -, !, \sim, \&, *$

$(- \text{ описание } -)$

идент

целое

веществ

строка

sizeof  $($   $\begin{cases} \text{выр} \\ \text{описание} \end{cases}$   $)$

$(- \text{ выр } -)$

$++, --$

$[ - \text{ выр } - ]$

$\cdot$   $\begin{cases} \text{идент} \\ \text{выр} \end{cases}$

$->$

$( \begin{cases} \text{выр} \\ \text{выр} \end{cases} )$

$\text{выр}$

$\text{выр} - ? - \text{выр} - : - \text{выр}$

$\text{выр}$

# Выражения – приоритет операций

1	() [] -> .	Скобки, выбор поля
2	! ~ - + * & sizeof (тип)	Унарные, приведение типа, sizeof
3	* / %	Умножение
4	+ -	Сложение
5	<< >>	Битовый сдвиг
6	< <= > >=	Сравнения
7	== !=	Равенство

8	&	Побитовое AND
9	^	Побитовое XOR
10		Побитовое OR
11	&&	Логическое AND
12		Логическое OR
13	?:	Условное
14	= += -= *= /= %= &=  = ^= <<= >>=	Присваивания

# Выражения – приоритет операций (пример)

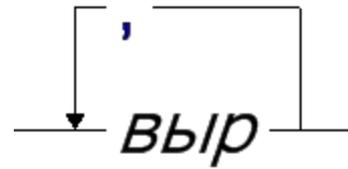
- $x = (*a[i+1].b + c * 2) \& ((x \& y | 07) \ll 3)$ 
  - [] и . сильнее, чем \*
  - \* сильнее, чем +
  - & сильнее, чем |
  - << сильнее, чем |
  - & сильнее, чем =
- Пусть `int a = 30000;` тогда
  - `((long) a * a)` равно 60000
  - `(long) (a * a)` – переполнение
  - `(long)` сильнее \*
- Скобки (в умеренном количестве) облегчают понимание

# Условные выражения

- Условное  $\text{--- выр ? --- выр : --- выр ---}$ 
  - Вычисляется только одна из ветвей  
 $i < 0 \parallel i \geq N \quad ? \quad 0 \quad : A[i]$
  - Тип – минимальный больший типов ветвей
- Логические связки (John McCarthy)
  - $A \ \&\& \ B$  эквивалентно  $(A \ ? \ B \ : \ 0)$
  - $A \ \parallel \ B$  эквивалентно  $(A \ ? \ 1 \ : \ B)$
  - Пример:  $i < N \ \&\& \ A[i] \ != \ 0$   
сравнить  $i < N \ \text{AND} \ A[i] \ <> \ 0$  (Pascal)
  - Пример:  $A[i] > \max \ \&\& \ A[i] = \max$  эквивалентно  
 $\text{if } (A[i] > \max) \ A[i] = \max;$

# Последовательное выражение

- Осмысленно только для выражений с побочным эффектом



- Результат – значение последнего выражения
- Пример:  
 $c = (a=3, b=2+a, a+b)$   
 $A[i,j] = i+j$  эквивалентно  $A[j] = i+j$

# Операторы

- Синтаксис:

;  
- пустой,  
ничего не  
делать

*выр*;  
- вычислить  
выражение и  
забыть.



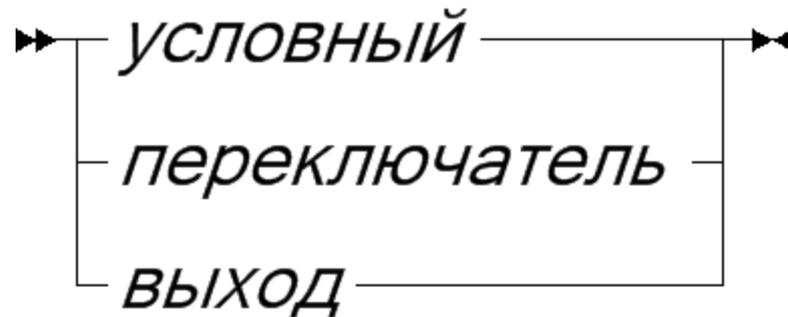
# Операторы - пример

```
y = (x = 1)+1;
loop:
  x += 2;
  goto swap;
  y = y++ + ++x;
swap:
  {
    int t;
    t = x, x=y, y=x;
    goto loop;
  }
x = 0;
```

- «Спагетти» код.
- Область видимости метки - по тем же правилам, что и для других объектов (?)
- Переменная t доступна только внутри блока
- В программе может быть **недоступный код**
- **goto** следует использовать только в крайних случаях.

# *Ветвления & C°*

- Выбор одной из ветвей в зависимости от значения выражения (условия)
- Синтаксис:



# Условный

- Синтаксис:

▶ *if* – ( – *выр* – ) – *оператор* →  
└─ *else* – *оператор* ─┘

- Пример:

C:

```
if ('a' <= c && c <= 'z'  
|| 'a' <= c && c <= 'п')  
    c -= 0x20;  
else if ('п' <= c && c <= 'я')  
    c -= 0x50;
```

VB:

```
If "a" <= c And c <= "z" _  
Or "a" <= c And c <= "п" Then  
    c = Chr(Asc(c)-&H20)  
Elseif "п" <= c And c <= "я" Then  
    c = Chr(Asc(c)-&H50)  
End If
```

# Условный оператор или условное выражение?

- Сравнить

```
if ('a' <= c && c <= 'z'  
    || 'a' <= c && c <= 'п')  
    c -= 0x20;  
else if ('p' <= c && c <= 'я')  
    c -= 0x50;
```

```
c = ('a' <= c && c <= 'z'  
     || 'a' <= c && c <= 'п'  
     ? c - 0x20  
     : 'p' <= c && c <= 'я'  
     ? c - 0x50  
     : c);
```

- Операторы внутри выражений (Algol-68)  
(вещ s:=0; для i = 1 до N цк s+:= A[i] кц; s)

# Условный оператор или условное выражение?

- Сравнить

```
if (x>0)
  if (y>0)
    *p ++;
  else
    *p --;
else
  if (y>0)
    *q ++;
  else
    *q --;
```

```
*(x>0 ? p : q) +=
  (y>0 ? 1 : -1)
```

- Условный оператор не вычисляет значения (а в Algol-68, Автокоде Эльбрус – вычисляет).

# Арифметический условный (FORTRAN)

C:

```
L = 0, R = N-1, found = 0;
Loop:
if (L<=R)
{
  i = (L+R) / 2;
  if (A[i] == x)
  {
    found = 1;
    goto Done;
  }
  else if (A[i] < x)
    R = i-1;
  else
    L = i + 1;
  goto Loop;
}
Done:
```

```
L = 0
R = N-1
FOUND = 0
100 IF (L-R) 101,101, 200
101 I = (L+R) / 2
    IF (A(I) - X) 102, 103, 104
102 R = I - 1
    GO TO 100
103 L = I + 1
    GO TO 100
104 FOUND = 1
200
```

# Переключатель

- Синтаксис:

▶ **switch** *( - Выр - )* →

выход:

▶ **break** *- ;* ▶

▶ { **case** *- Выр - :* оператор } ▶  
▶ **default** *- :* ▶

- Семантика:

- Переход на метку **case**, соответствующую значению выбирающего выражения
- При отсутствии таковой – переход на метку **default**
- При отсутствии как нужного **case**, так и **default** – переход на конец переключателя
- **break;** - переход на конец переключателя.

# Переключатель – пример (C)

```
switch (x % 6)
{
    case 0 :
    case 2:
        x += 2;
    default :
        x += 1;
        break;
    case 1 :
        x = 0;
        break;
}
```

```
int t = x % 6;
if (t==0) goto L0
if (t==1) goto L1;
if (t==2) goto L2;
goto Ldefault;
L0 :
L2 :      x += 2;
Ldefault : x += 1;
          goto Ldone;
L1 :      x = 0;
          goto Ldone;
Ldone:
```

# Переключатель – пример (C)

```
int t = x % 6;
if (t==0) goto L0
if (t==1) goto L1;
if (t==2) goto L2;
goto Ldefault;
L0 :
L2 :      x += 2;
Ldefault : x += 1;
          goto Ldone;
L1 :      x = 0;
Ldone:
```

- $x \% 6$  вычисляется один раз;
- Выбор метки можно реализовать эффективнее (таблица, дихотомия,...)
- «Провал» после  $x += 2$ ; - важен порядок ветвей
- Значения case - константы

# Переключатель – пример (Pascal)

```
switch (x % 6)
{
    case 0 :
    case 2:
        x += 2;
    default :
        x += 1;
        break;
    case 1 :
        x = 0;
        break;
}
```

```
case x mod 6 of
0, 2: begin
        x := x+2;
        x := x+1;
    end;
1 : x := x+1;
else
        x := 0;
end
```

# Переключатель – пример (Pascal)

```
case ch of  
'A'..'Z', 'a'..'z' :  
    WriteLn('Буква');  
'0'..'9' :  
    WriteLn('Цифра');  
'+', '-', '*', '/' :  
    WriteLn('Оператор');  
else  
    WriteLn('Спецсимвол')  
end
```

Интервалы  
значений в  
альтернативных:

- Наглядность
- Возможность транслятору эффективно реализовать переход

# Переключатель – пример (Visual Basic)

## Select Case True

**Case** n = AscW("<")

t = t & "&lt;";

**Case** n = AscW(">")

t = t & "&gt;";

**Case** n = AscW("&")

t = t & "&amp;";

**Case** n = AscW("'")

t = t & "&apos;";

**Case** n = AscW("''")

t = t & "&quot;";

**Case** n > 254, n < 0

t = t & "&#x"

& Hex(n And &HFFFF) & ";"

**Case Else**

t = t & Chr(n)

**End Select**

- Выражения в Case – не обязательно константы
- Последовательный выбор альтернативы
- Полностью эквивалентно If ... Elseif... Else ... End If

# Переключатель – пример (Fortran)

```
int t = x % 6;  
if (t==0) goto L0  
if (t==1) goto L1;  
if (t==2) goto L2;  
goto Ldefault;  
L0 :  
L2 :      x += 2;  
Ldefault : x += 1;  
          goto Ldone;  
L1 :      x = 0;  
Ldone:
```

```
t = X - (X/6)*6 +1  
GOTO (0,1,2,3,3,3) t  
0  CONTINUE  
2  X = X+2  
3  X = X+1  
   GO TO 100  
1  X = 0  
100
```

(CONTINUE – пустой оператор)

# Вычисляемые метки (Fortran)

```
t = X - (X/6)*6 + 1
GOTO (0,1,2,3,3,3) t
0  CONTINUE
2  X = X+2
3  X = X+1
   GO TO 100
1  X = 0
100
```

0,1,2,4,100 – метки

0,1,2,6 - константы

```
t = X - (X/6)*6
ASSIGN 3 TO L
IF (t .EQ.0) ASSIGN 0 TO L
IF (t .EQ.1) ASSIGN 1 TO L
IF (t .EQ.2) ASSIGN 2 TO L
GOTO L, (0,1,2,3)
0  CONTINUE
2  X = X+2
3  X = X+1
   GO TO 100
1  X = 0
100
```

# Циклы & C°

Синтаксис:

► **for** – ( выр ; выр ; выр ) – оператор ►

– **break** – ;

– **continue** – ;

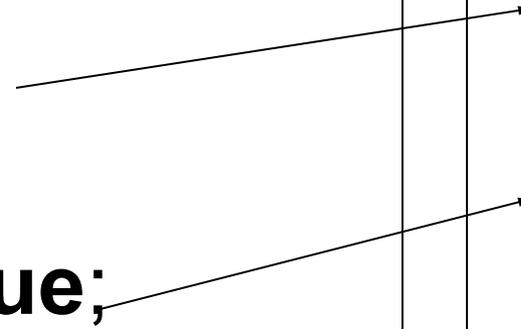
– **while** – ( – *выр* – ) – оператор

– **do** – оператор – **while** – ( – *выр* – ) – ;

# Базовая форма цикла

```
for (;;)
{
    ...
break;
    ...
continue;
    ...
}
```

```
Loop:
{
    ...
goto Done;
    ...
goto Loop;
    ...
goto Loop;
}
Done:
```



# Цикл for

```
for (Init-expr;  
      Test;  
      Reinit-expr)  
{  
    ...  
}
```

```
Init-expr;  
for (;;)   
{  
    if (! Test)  
        break;  
    ...  
    Reinit-expr;  
}
```

# Циклы `while` и `do...while`

```
while (Test)
```

```
{
```

```
...
```

```
}
```

```
for (;Test;)
```

```
{
```

```
...
```

```
}
```

```
do
```

```
{
```

```
...
```

```
} while (Test);
```

```
for (;;)
```

```
{
```

```
...
```

```
  if (!Test) break;
```

```
}
```

# Другие формы цикла

Альфа-6	N раз ...	for (k=N; k--;) ...
Pascal	repeat ... until x<0.01	do ... while (x>=0.01);
Algol-60	for x:=0.2 step 0.01 until 1.0 do ...	for (x=0.2; x<=1.0; x+=0.01) ...
Algol-60	for i=1,2,-3,4,10 do ...	int is[] = {1,2,-3,4,10}; for (k=0; i=is[k], k<5; k++) ...
Algol-60	for i=1, 4 while x<0.01, N step -1 until 10 do ...	i=1; ... for (i=4; x<0.01; ) ... for (i=N; i>=10; i--) ...
Visual Basic	Dim A(1 To N) As Integer For Each x In A ... x ... Next x	int A[100], *x, i; for (*x=A, i=100; i--; x++) ... *x ...
SETL	print({n in {2..N}   forall m in {2..n - 1}   n mod m > 0});	

# Циклы - SETL

```
print({n in {2..N} |  
  forall m in {2..n - 1} | n mod m > 0});
```

```
for (int n=2; n<=maxN; n++)  
{  
  int test = 0, m;  
  for (test=0, m=2; m<=n-1 && (test=n % m); m++);  
  if (test)  
    printf(“%d “, n);  
}
```

# Переменная цикла

- Pascal – значение переменной после выполнения цикла неопределено:

```
for i:=1 to N do  
  if A[i] = 0 then break;  
  WriteLn(i);
```

- Algol-68 – переменная цикла является константой в теле цикла

```
for i from 1 to N do  
  if A[i] = 0 then i:=N+1 fi  
od
```

- C – допускается определение переменной в заголовке цикла

```
for (int i=0; i<N; i++) printf(“%d\n”,i+=2);
```

# Границы цикла

- Алгол-68, Visual Basic – вычисляются один раз
- С – вычисляется каждый раз, поскольку понятия «границы цикла» нет

```
int L = strlen(s), i;
```

```
char * p = s;
```

```
for (i=0,p=s; i<L; p++)
```

```
    *p == 'a' ? (strcpy(p,p+1), L--) : i++;
```

# Помеченные циклы

Проблема: **break** и **continue** действует на ближайший охватывающий цикл или переключатель

Пример: первая строка матрицы с нулевым элементом

```
for (i=0; i<n; i++)  
{  
    found = 0;  
    for (j=0; j<N; j++)  
        if (A[i][j] == 0)  
            {  
                found = 1;  
                break;  
            }  
    if (found) break;  
}
```

Неправильно:

```
for (i=0; i<n; i++)  
    for (j=0; j<N; j++)  
        if (A[i][j] == 0)  
            break;
```

Java:

```
iloop:  
for (i=0; i<n; i++)  
    for (j=0; j<N; j++)  
        if (A[i][j] == 0)  
            break iloop;
```

```
for (i=0; i<n; i++)  
    for (j=0; j<N; j++)  
        if (A[i][j] == 0)  
            goto done;  
done:
```