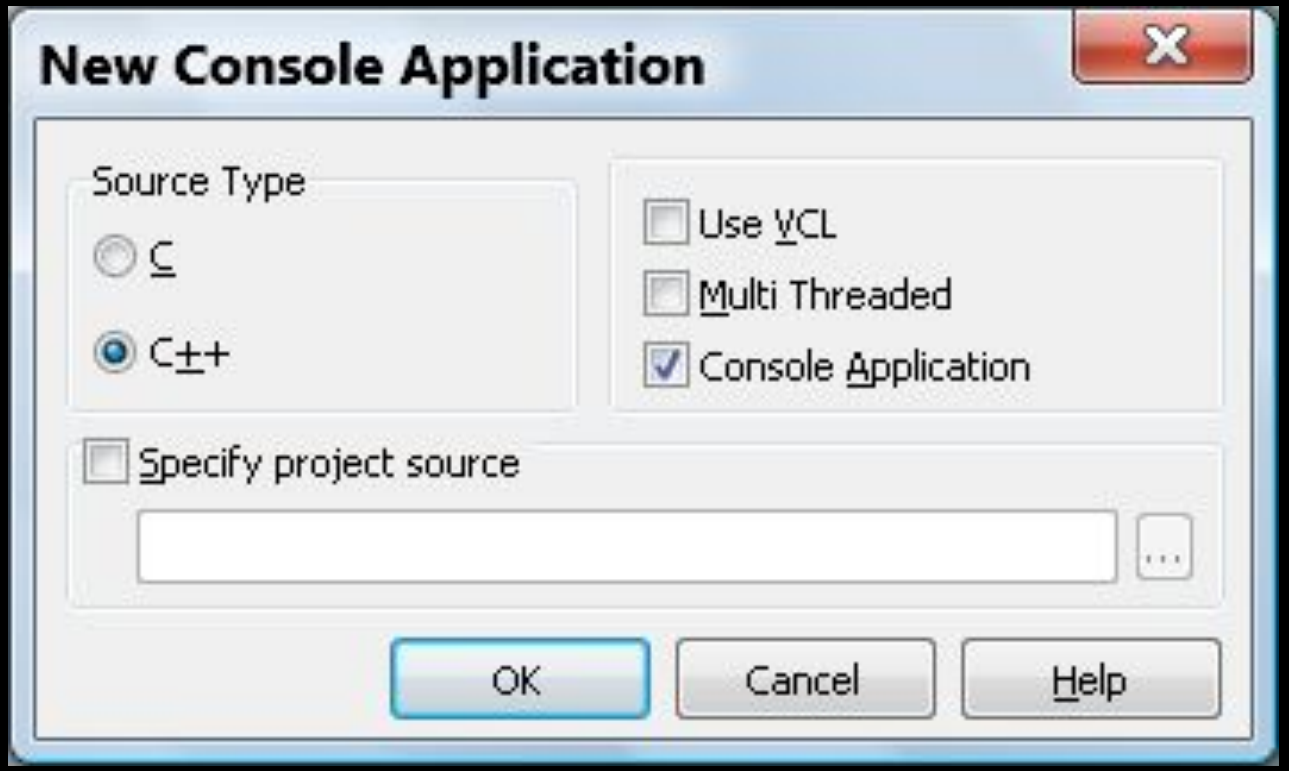
Быстрое введение в язык C++

C++ Builder – консольное приложение

либо Turbo C

Гречкина П.В., ПЯВУ-2,
C++

Создание консольного приложения



Структура программы

```
//---------------------------------------------------------------------
-

#pragma hdrstop

//---------------------------------------------------------------------
-

#pragma argsused
int main(int argc, char* argv[])
{
        return 0;
}
//---------------------------------------------------------------------
-
```

```
int main()
{
        return 0;
}
```

```
void main()
{
        return;
}
```

Структура программы

C++ Builder

```
//---------------------------------------
int main()
{
        return 0;
}
//---------------------------------------
```

```
//---------------------------------------
void main()
{
        return;
}
//---------------
```

Delphi

```
Program Prog1;
{$APPTYPE CONSOLE}
//---------------------------------------
function main : integer;
begin
        Result := 0; exit;
end;
//--------
Begin
        mai
End.
```

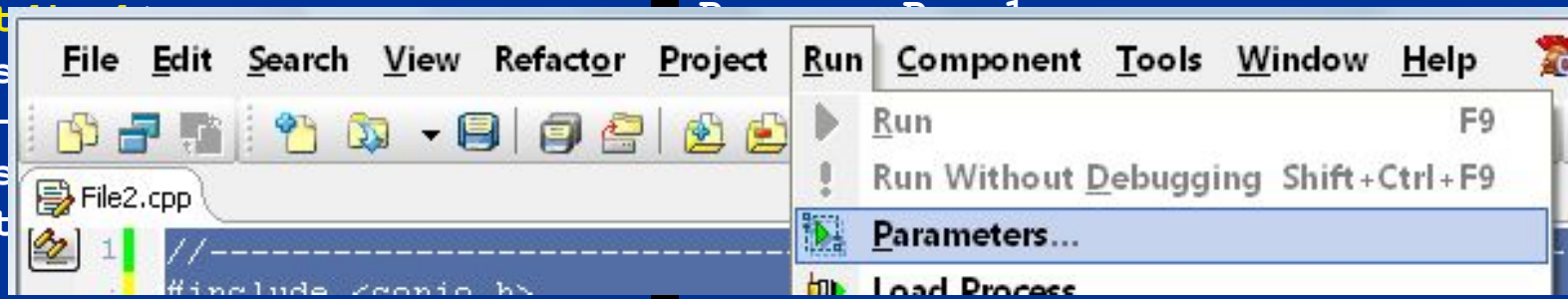
```
Program Prog2;
{$APPTYPE CONSOLE}
Begin

End.
```

```
Program Prog2;
{$APPTYPE CONSOLE}
//---------------------------------------
procedure main;
begin
        exit;
end;
//---------------------------------------
Begin
        main;
End.
```

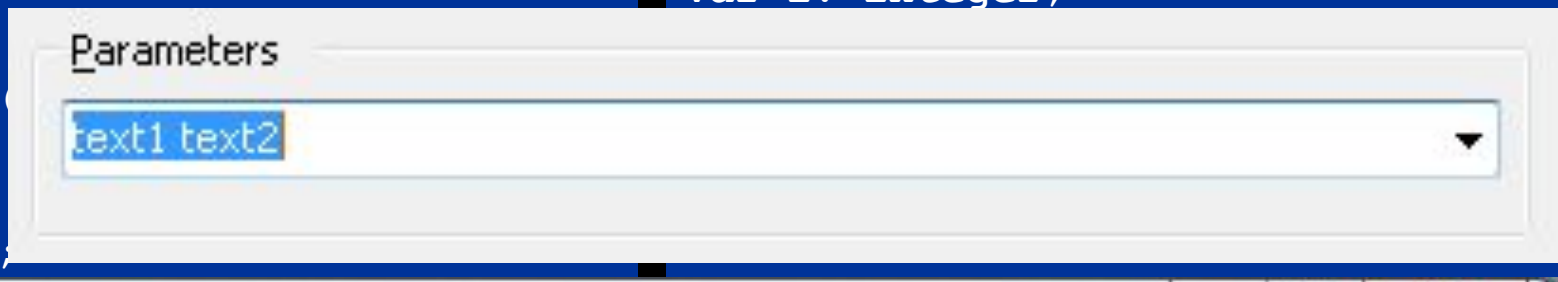
Параметры программы

```
//---------------------------------
#include <conio.h>
#include <st
#pragma hdrs
//---------
#pragma args
int main(int
{
    int i;
    for (i =
      printf
    }
    getch();
    return 0;
```

Меню Run -> Parameters…




```
var i: integer;
```



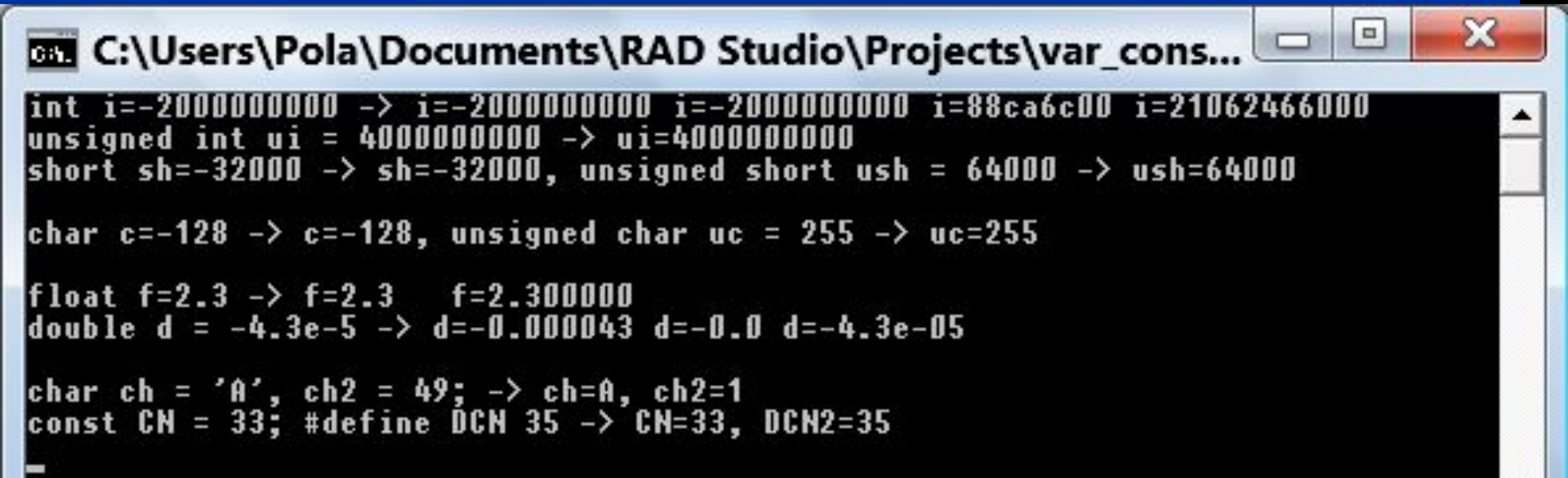
Простые

```
Const
    CN=33;
    DCN =35;
Var
    i: integer = -2000000000;  // 4 байта
    ui: cardinal = 4000000000;
    sh: Smallint = -32000;  // 2 байта
    ush: Word = 64000;
    c: ShortInt = -128;  // 1 байт
    uc: Byte = 255;
    f: Single = 2.3; // float point - 4 байта
    d: Real = -4.3e-5; // float point 8  = double
    ch: char = 'A';
    ch2: char = #49; // «1»
```

```
//------------------
#include <conio.h>
#include <stdio.h>
#pragma hdrstop
//------------------
const CN = 33;
#define DCN 35
//------------------
int main() {
    int i=-2000000000;  unsigned int ui = 4000000000;    // 4 байта

    short sh=-32000; unsigned short ush = 64000;  // 2 байта

    char c=-128; unsigned char uc = 255;       // 1 байт

    float f=2.3; double d = -4.3e-5;  // с плавающей точкой

    unsigned char ch = 'A', ch2 = 49; // символ  1 байт
…
```

Простые типы

```
...
  printf("int i=-2000000000 -> i=%li i=%d i=%x i=%o \n", i,i,i,i);
  printf("unsigned int ui = 4000000000 -> ui=%u\n", ui);
  printf("short sh=-32000 -> sh=%d, ", sh);
  printf("unsigned short ush = 64000 -> ush=%d \n\n", ush);
  printf("char c=-128 -> c=%d, ", c);
  printf("unsigned char uc = 255 -> uc=%d \n\n", uc);

  printf("float f=2.3 -> f=%3.1f   f=%f \n", f, f);
  printf("double d = -4.3e-5 -> d=%lf d=%3.1lf d=%g\n\n", d, d, d);

  printf("unsigned char ch='A', ch2=49; -> ch=%c, ch2=%c \n", ch, ch2);
```



Операторы

Таблица 2.1. Операторы присвоения, сравнения и логические

Оператор	Object Pascal	C	Visual Basic
Присвоение	:=	=	=
Сравнение	=	==	= или is*
Неравенство	<>	!=	<>
Меньше, чем	<	<	<
Больше, чем	>	>	>
Меньше или равно	<=	<=	<=
Больше или равно	>=	>=	>=
Логическое *и*	and	&&	and
Логическое *или*	or	||	or
Логическое *нет*	not	!	not

*Оператор сравнения is используется только для объектов, тогда как оператор сравнения = применяется для всех остальных типов данных.

C++: int i=j=0; // не путать с i=j==0

if (i==j) // не путать с if (i=j)

Операторы

Таблица 2.2. Арифметические операторы

Оператор	Object Pascal	C	Visual Basic
Сложение	+	+	+
Вычитание	-	-	-
Умножение	*	*	*
Деление с плавающей точкой	/	/	/
Деление целых чисел	div	/	\
Деление по модулю	mod	%	Mod
Возведение в степень	Отсутствует	Отсутствует	^

```
y=1/3*x;  // будет 0
y=1.0/3*x;
```

```
Int x=1, z=3;  y= x / z; // будет 0
y=x / (double)z;
```

Операторы

Таблица 2.3. Побитовые операторы

Оператор	Object Pascal	C	Visual Basic
И	and	&	And
Не	not	~	Not
Или	or	|	Or
Исключающее или	xor	^	Xor
Сдвиг влево	shl	<<	Нет
Сдвиг вправо	shr	>>	Нет

```
5 & 6 => 4      5 | 6 => 7      5 ^ 6 => 3
0101            0101            0101
0110            0110            0110
0100            0111            0011
```

```
+=  -=  *=  /=  %=
&=  |=  ^=  <<=  >>=
```

Операторы

Процедуры увеличения и уменьшения

Процедуры увеличения и уменьшения генерируют оптимизированный код для добавления или вычитания единицы из целой переменной. Object Pascal не предоставляет таких широких возможностей, как постфиксные и префиксные операторы ++ и -- в языке C. Тем не менее процедуры `Inc()` и `Dec()` обычно компилируются в одну команду процессора.

Delphi (фрагмент)

```
Var k: integer; pk: ^integer;

Begin
K:=10;
Inc( k );  // k := integer(Ord(k)+1);
Dec( k );  // -1

Inc( k, 2);  //+2
Dec( k, 3); //-3

AllocMem( pk, 2* SizeOf(integer));
Inc( pk ); // + SizeOf(integer);
Dec( pk ); FreeMem(pk, 2* SizeOf(integer));
```

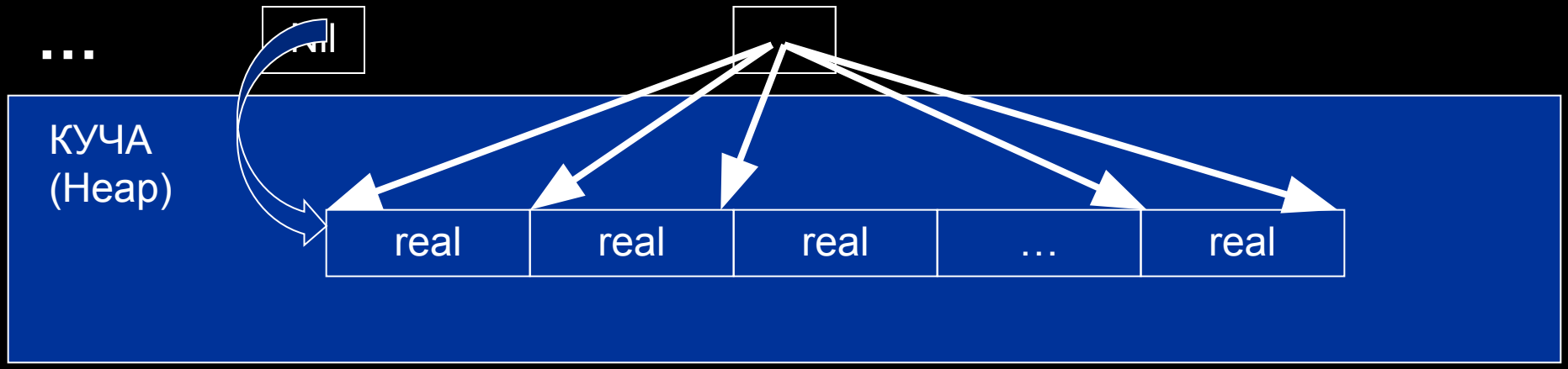
C++ (фрагмент)

```
int k=10, *Pk;
k++;   --k;

k= k + ++k;  printf("%d\n", k);  //22
k=10; k= sum (k, ++k); printf("%d", k); //22
k=10; k= sum (k, k++); printf("%d", k); //21
k=10;
k= sum (++k, ++k); printf("%d", k); //23
k=10;
k= sum (k++, k++); printf("%d", k); //21
Pk = (int*) calloc(2, sizeof(int)) ;
Pk++;  Pk--;
free (Pk);
```

```
int sum (int k1, int k2)
{
        return k1+k2;
}
```

Динамический массив (свой II)

```
Type PReal = ^Real;
Var DynAr: PReal;  Cur: PReal;//текущий эл-т
…
```




```
GetMem( DynAr, N*SizeOf( real ));
Cur:= DynAr; // на начало
For i:=0 to N-1 do
begin
    ReadLn(Cur^ );      Inc(Cur);
end;
```

Массивы

Массивы

Object Pascal позволяет создавать
Например, ниже объявляется перемен
чисел.

```
var
  A: Array[0..7] of Integer;
```

Такой оператор эквивалентен следующему объявлению в языке C:

```
int A[8];
```

```
Var   mas: array of Integer;
Begin
  SetLength( mas, 8 );
  for i := 0 to 7 do begin
    mas[i] = (i-2)*sqr(i-2);
    write(i-2,'^3=',mas[i],' ');
  end;
  mas:=nil;
End;
```

```
  int *mas;
  mas = new int[8];
  for (i = 0; i < 8; i++) {
    mas[i] = pow(i-2,3);  // math.h
    printf("%d^3=%d ", i-2, mas[i]);
  }
  delete [] mas;
```

```
-2^3=-8 -1^3=-1 0^3=0 1^3=1 2^3=8 3^3=27 4^3=64 5^3=125
```

C++

Записи - Структуры

Записи

Структуры, определяемые пользователем, в Object Pascal называются *записями* (record). Они эквивалентны типам данных struct в языке C или Type — в языке Visual Basic:

```
{ Pascal }
Type
  MyRec = record
    i: Integer;
    d: Double;
  end;

/* C */
typedef struct {
  int i;
  double d;
} MyRec;
```

Записи с вариантами

Object Pascal также поддерживает *вариантные записи* (*variant records*), которые обеспечивают хранение разнотипных данных в одной и той же области памяти. Не путайте эти записи с рассмотренным выше типом `Variant` — вариантные записи позволяют независимо получать доступ к каждому из перекрывающихся полей данных. Если вы знакомы с языком C или C++, то можете понимать вариантные записи как аналог концепции `union` в структурах языка C или C++. Приведенный ниже код показывает вариантную запись, в которой поля типа `Double`, `Integer` и `Char` занимают одну и ту же область памяти.

```
type
  TVariantRecord = record
    NullStrField: PChar;
    IntField: Integer;
    case Integer of
      0: (D: Double);
      1: (I: Integer);
      2: (C: char);
  end;
```

```
Var
      h: TVariantRecord;
Begin
      h.IntFielf := 6;
      h.c := 'T';
```

```
typedef struct {
     char* StrField;
     int IntFielf;
     union {
     double d;
     int I;
     char c;
     };
} TVariantRecord;


     TVariantRecord h;
     h.IntFielf = 6;
     h.c = 'T';
```

Операторы условного перехода

Оператор условного перехода if

Оператор if позволяет проверить, выполняется ли некоторое условие, и, в зависимости от результатов этой проверки, выполнить тот или иной блок кода. В качестве примера ниже приведен простейший фрагмент кода с использованием оператора if и его аналоги на языках C и Visual Basic:

```
{ Pascal }
if x = 4 then y := x;
/* C */
if (x == 4) y = x;
```

```
y= (x==4 ? x : y)
```

Условный переход

```
if (4==x)  y=x;
else y=fabs(x);
```

```
y= ( 4==x ? x : fabs(x) );      // math.h
```

```
If (4==x) { y=x;}
else { y=fabs(x); x=-x; }
```

```
y= ( 4==x ? x : x=-x, fabs(x) );
```

```
if (4==x) {
 y=x;
}
else {
y=fabs(x); x=-x;
}
```

Следование “,”

Множественный выбор

Оператор case

Оператор case в языке Pascal подобен операторам switch в языках C или C++. Он позволяет сделать выбор одного из нескольких возможных вариантов без использования сложных конструкций, состоящих из нескольких вложенных операторов if else. Вот пример выполнения такого выбора:

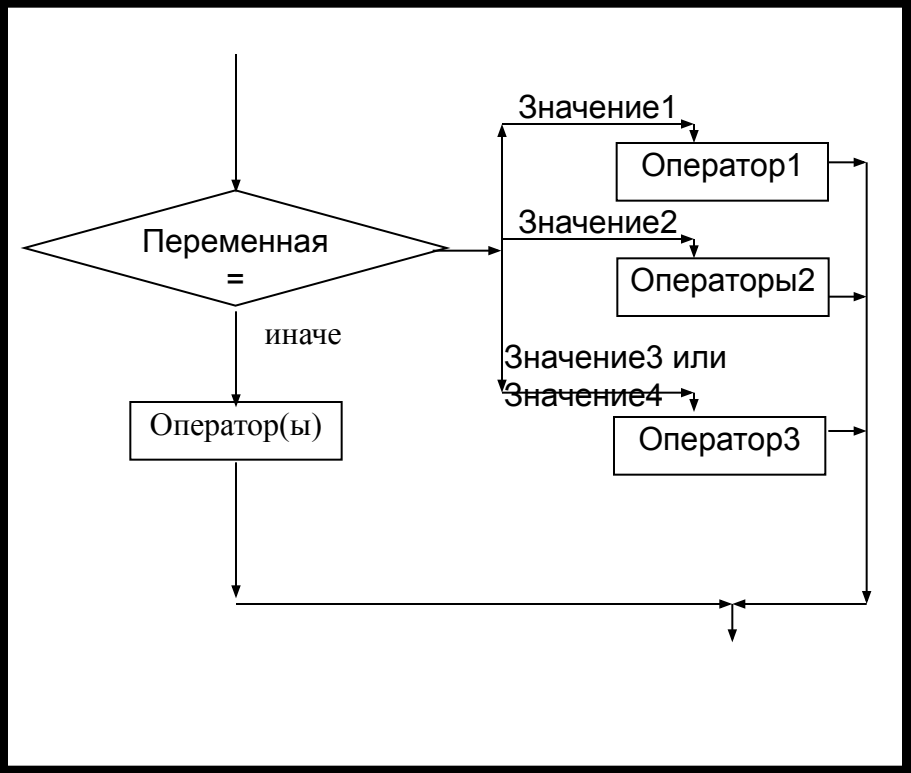
```
case SomeIntegerVariable of
  101 : DoSomething;
  202 : begin
      DoSomething;
      DoSomethingElse;
    end;
  303 : DoAnotherThing;
  else DoTheDefault;
end;
```

А вот как выглядит этот же пример на языке C:

```
switch (SomeIntegerVariable)
{
  case 101: DoSomeThing; break;
  case 202: DoSomething;
            DoSomethingElse; break
  case 303: DoAnotherThing; break;
  default : DoTheDefault;
}
```

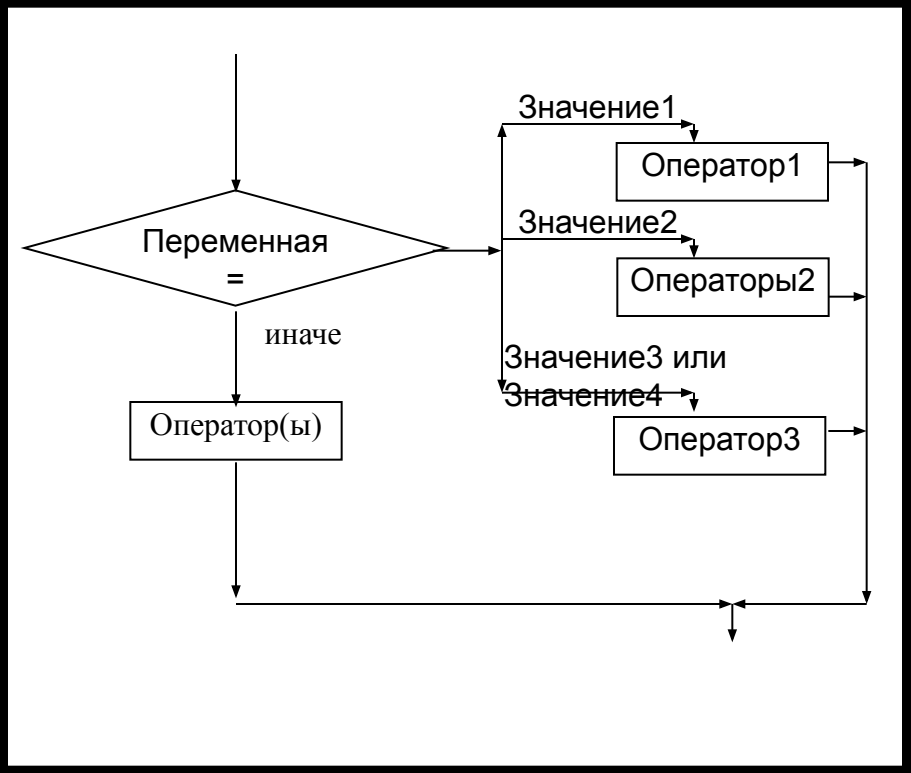
На В операторе case управляющая переменная должна иметь перечислимый тип. В ча-

Множественный выбор в Delphi



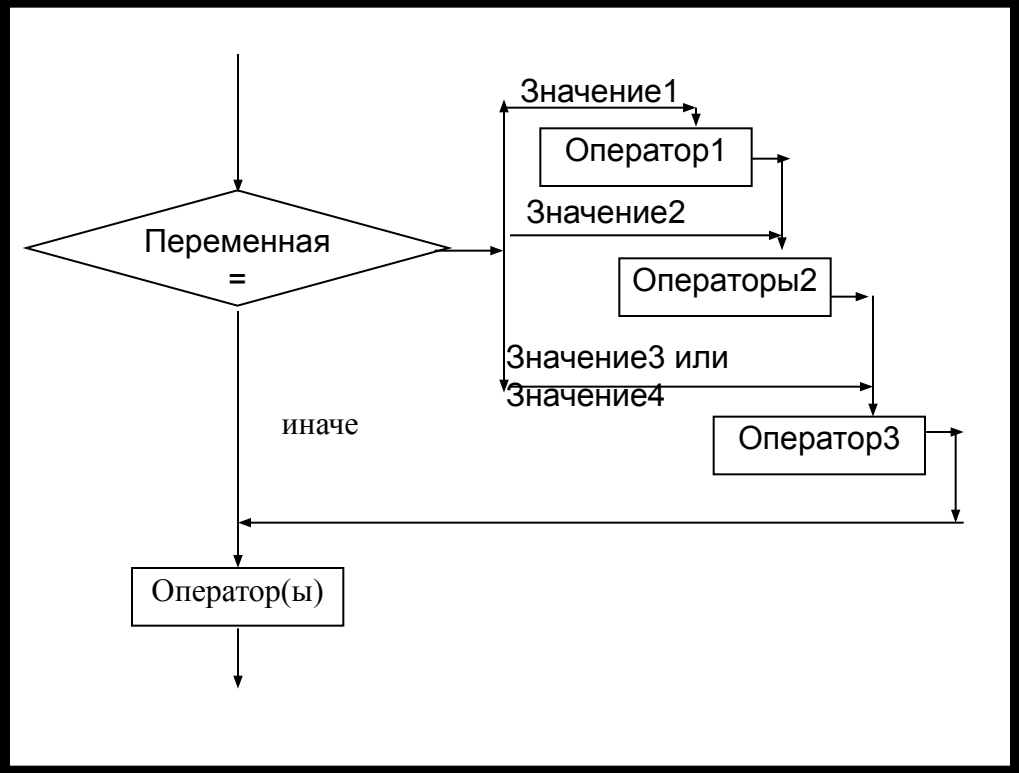

```
Case Переменная of
    Значение1: Оператор1;
    Значение2: begin
                    Оператор2_1;
                    . . . . . .
                    Оператор2_K;
                end; {Значение2}
    Значение3, Значение4: Оператор3;
    else
        begin
            Оператор_1;
            . . . . . .
            Оператор_R;
        end; {else}
End; {Case}
```

Множественный выбор в Си




```
switch ( Переменная ) {
   case Значение1: Оператор1; break;
   case Значение2:
                        Оператор2_1;
                        . . . . . .
                        Оператор2_K;
                        break;
   case Значение3, Значение4:
      Оператор3;
            break;
   default:
                  Оператор_1;
                  . . . . . .
                  Оператор_R;
}
```

Множественный выбор в Си




```
switch ( Переменная ) {
   case Значение1: Оператор1;
   case Значение2:
                        Оператор2_1;
                        . . . . . .
                        Оператор2_K;

   case Значение3, Значение4:
                   Оператор3;

   default:
                  Оператор_1;
                  . . . . . .
                  Оператор_R;
}
```

Цикл FOR

Цикл for

Цикл for больше всего подходит для организации повторения некоторых действий заранее известное число раз. Например, ниже приведен текст цикла for (хотя и не слишком полезного), в котором к значению переменной десять раз прибавляется счетчик цикла (дальше следуют аналоги этого цикла на языках C

```
{ Pascal }
var
  I, X: Integer;
begin
  X := 0;
  for I := 1 to 10 do
    inc(X, I);
end.
```

```
/* C */
void main(void) {
  int x, i;
  x = 0;
  for(i=1; i<=10; i++)
    x += i;
}
```

```
Var X,I: single;
Begin
X:=0;  I:=1;
While I<=2.05 Do
Begin
      X:=X+I;
      I:=I+0.1
End;
```

```
float x=0, i;
for (i=1.0 ; i<=2.05; i+=0.1){
      x+=i;
}
```

C++

Следование “,”

Цикл FOR

```
float x=0, i;
for (i=1.0 ; i<=2.05; i+=0.1){
      x+=i;
}
```

```
float x, i;
for (x=0,i=1.0 ; i<=2.05; x+=i, i+=0.1);
```

```
float x=0, i=1.0;
for (  ; i<=2.05; ) x+=i, i+=0.1;
```

```
float x=0, i=1.0;
for (  ; i<=2.05; ) {x+=i; i+=0.1;}
```

```
float x=0, i=1.0;
while (i<=2.05) {
      x+=i; i+=0.1;
}
```

Циклы

while усл do ;

```
while (усл) { }
for ( ; усл ; ) { }
```

repeat
until усл;

```
do { }
while (!усл);
```

break;

```
break;
```

continue;

```
continue;
```