[illegible] [illegible]

[illegible]
[illegible]

[illegible] :
[illegible]., [illegible]., [illegible]. [illegible]. - [illegible], 1975. - [illegible] 7 - 97.

[illegible]

1. [illegible]
2. [illegible]
3. [illegible]
4. [illegible]
5. [illegible]
6. [illegible]
7. [illegible]
8. [illegible]
9. [illegible]
10. [illegible]
11. [illegible]
12. [illegible]
13. [illegible]
14. [illegible]
15. [illegible]
16. [illegible]
17. [illegible]

Ѧ ɖ ɔ ɑ ɔ ɔ ɜ ɔ ɜ ɔ ɔ ɜ ɔ ɔ ɜ ɔ

0 ⌐⌐⌐

Ꞇ+ ɔɛɛɛɔ ɔɛɛɛɔ ɔɛɛɛɔ

[illegible]

[illegible]

4. [illegible]

[illegible]

[illegible]

[illegible]

(1) [illegible];
(2) [illegible];
(3) [illegible].

4.1. [illegible]

[illegible]

[illegible]:

```
dd:=dd/2;
if dd ≤ r do r:=r-dd
```

[illegible] r [illegible] dd [illegible]

$$0 \le r < dd \quad (1)$$

[illegible] (1) [illegible] dd, [illegible] r, [illegible]

$$0 \le r < 2*dd \quad (2)$$

[illegible].

1. $dd \le r$. [illegible] (2) [illegible]

$$dd \le r < 2*dd \quad (3)$$

[illegible] **do**. [illegible] r [illegible] dd, [illegible] (3) [illegible]

$$0 \le r < dd$$

[illegible] (1) [illegible].

2. [illegible] $dd \le r$ ([illegible] $dd > r$).

[illegible] **do** [illegible] r. [illegible] $dd > r$ [illegible] (2), [illegible]

$$0 \le r < dd$$

[illegible] (1).

[illegible] (1); [illegible].

4.2. [illegible]

[illegible]

[illegible] $d_0, d_1, d_2, d_3, \ldots$

[illegible]:

```
    i=0 d_i=D              (2a)
    i>0 d_i = f(d_{i-1})   (2b)
```

[illegible] D - [illegible] f - [illegible] d [illegible] d_k [illegible] k. [illegible]

$$d=d_k \quad (3)$$

[illegible] k [illegible]

$$[illegible](d_k) \quad (4)$$

[illegible]

$$\text{non } [illegible](d_i) \quad (5)$$

[illegible] i, [illegible] $0 \le i < k$

[illegible]:

```
d:=D
while non [illegible] (d) do d:=f(d)     (6)
```

[illegible] **while ... do**. ([illegible] **if ... do**, [illegible]

```
while B do S
```

[illegible]

```
if B do
begin S; while B do S end
```

[illegible] **non** [illegible].)

[illegible] **while** B **do** S [illegible] S [illegible] (6) [illegible] n-[illegible] ([illegible] $n \ge 0$)

$$d=d_n \quad (7a)$$

[illegible]

$$\text{non } [illegible](d_i) \text{ [illegible] } i, [illegible] \ 0 \le i < n \quad (7b)$$

[illegible] $n=0$; [illegible] ([illegible]), [illegible] $n=N$ $(N \ge 0)$, [illegible] $n=N+1$.

[illegible] N-[illegible] (7a) [illegible] (7b) [illegible] $n=N$. [illegible] $(N+1)$-[illegible]

$$\text{non } [illegible](d)$$

[illegible] (7a) [illegible] $n=N$ ([illegible] $d=d_N$) [illegible]

$$\text{non } [illegible](d_N)$$

[illegible] (7b) [illegible] $n=N+1$. [illegible] $d=dN$ [illegible] (2b) [illegible]

$$f(d)=d_{N+1}$$

[illegible] $(N+1)$-[illegible]

$$d:=f(d)$$

[illegible] (7a) [illegible] $N=N+1$, [illegible] (7) [illegible].

[illegible] k-[illegible] $n>k$ [illegible] n-[illegible] (7b) [illegible]

$$\text{non } [illegible](d_k)$$

[illegible] (4).

[illegible] n-[illegible]

$$\text{non}(\text{non } [illegible](d))$$

[illegible] (7a) [illegible]

$$[illegible](d_n) \quad (8)$$

[illegible] $n<k$, [illegible] (5). [illegible] $n=k$, [illegible] (3) [illegible] (7a), (4) [illegible] (8) [illegible] (5) - [illegible] (7b). [illegible]

[illegible] (6) [illegible] (3), (4) [illegible] (5), [illegible] ...

[illegible]

[illegible] (!), [illegible] ([illegible].)

[illegible] (6) [illegible] (6) [illegible]

[illegible]

[illegible]

4.3. [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

5. [illegible]

[illegible]:

```
integer  r,  dd;
r:=a; dd:=d;
while dd ≤ r do dd:=2*dd;
while dd ≠ d do
begin dd:=dd/2;
    if dd ≤ r do r:=r-dd;
end
```

[illegible] a [illegible] d [illegible]

$$a \geq 0 \quad [illegible] \quad d \geq 0$$

[illegible]

[illegible] $i=0$ $dd_i=d$

[illegible] $i>0$ $dd_i=2*dd_{i-1}$

[illegible]

$$dd_n=d*2^n \quad (1)$$

[illegible] $d>0$, [illegible] r [illegible]

$$dd_k > r$$

[illegible] k; [illegible]

$$dd=d*2^k$$

[illegible]

$$dd_i=2*d_{i-1}$$

[illegible] d_{i-1}, [illegible]

$$d_{i-1}=d_i/2$$

[illegible]

[illegible]

$$dd=dd_k$$

[illegible]

$$0 \leq r < dd \quad (2)$$

[illegible] (**while** $dd \neq d$ **do**) [illegible]

$$dd=d$$

[illegible] (2) [illegible]

$$0 \leq r < d \quad (3)$$

[illegible]

$$dd \equiv 0 \ \mathbf{mod}\ (d) \quad (4)$$

[illegible] dd [illegible] (1))

$$d*2^i \ [illegible]\ 0 \leq i \leq k$$

[illegible] r [illegible]

$$a \equiv r \ \mathbf{mod}\ (d) \quad (5)$$

(1) [illegible].

(2) [illegible] ($dd:=2*dd$) [illegible] (5), [illegible] (5).

(3) [illegible] ($dd:=dd/2$) [illegible] (5); [illegible] (5), [illegible] r, [illegible] r [illegible] dd, [illegible] (4) [illegible] (5). [illegible] (5), [illegible] (5).

[illegible] (3) [illegible] (5), [illegible] r [illegible]

$$0 \leq r < d \quad [illegible] \quad a \equiv r \ \mathbf{mod}\ (d)$$

[illegible] r [illegible] d.

[illegible] 1. [illegible]

```
integer r, dd, q;
r:=a;  dd:=d;   q:=0;
while dd ≤ r do dd:=2*dd;
while dd ≠d do
    begin dd:=dd/2; q:=2*q;
        if dd ≤ r do begin r:=r-dd; q:=q+1 end
end
```

[illegible] q [illegible]

$$a=q*dd+r$$

([illegible])

[illegible] 2. [illegible]

while [illegible] **do** S

[illegible] S [illegible]

$$P \wedge B$$

[illegible] S [illegible]

[illegible]

[illegible])

[illegible] 3. [illegible] z [illegible]

$$A > 0 \quad [illegible] \quad B > 0$$

[illegible]

```
y:=[illegible];  [illegible]:=[illegible];  z:=1;
while y ≠ 0 do
    begin if [illegible] ([illegible]) do
        begin y:=y -1; z:=z*x end;
        y:=y/2; x:=x*x;
    end
```

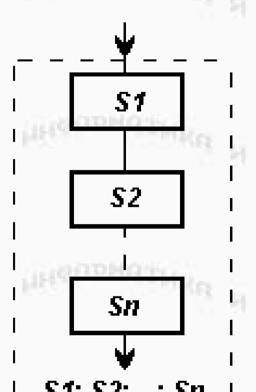
[illegible] z = [illegible]^B.

[illegible] ([illegible]:=[illegible]/2) [illegible]

$$x > 0 \quad [illegible] \quad y \geq 0 \quad [illegible] \quad [illegible]^B = z*[illegible]^y$$

4 + ⅆⅆⅇⅇⅆⅆⅇⅇⅆⅆⅇⅇ

7.



S1; S2; ...; Sn

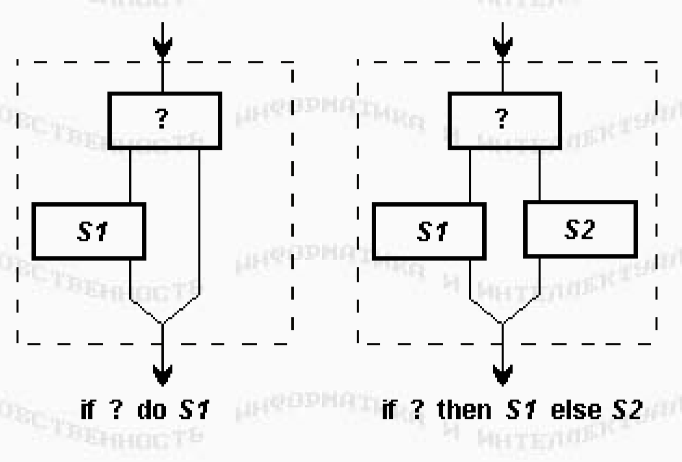
```
dd:=dd/2
if dd<= r do r:=r - dd
```

- dd;
- dd.

0<=r<dd

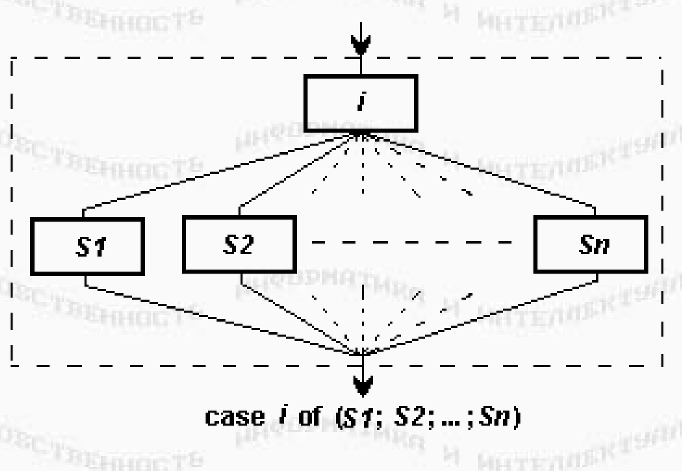
if dd<= r **do** r **-** dd

if ... **then** ... 1 **else** ... 2



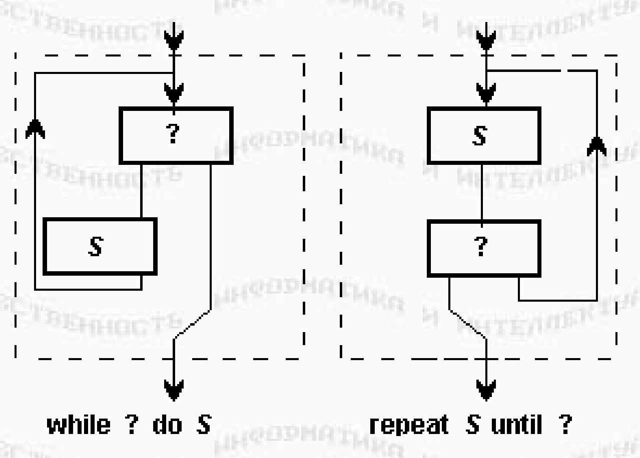
if ? do S1 | if ? then S1 else S2

case-of



case i of (S1; S2; ...; Sn)

"**while** ... **do** ..." "**repeat** ... **until** ..."



while ? do S | repeat S until ?

```
dd:=dd/2;
if dd^ r do r:=r-dd
```

0<=r<dd

8. [illegible]

[illegible]

[illegible]

[illegible]

[illegible] "B2" [illegible] "S1", [illegible] "S2"), [illegible]

```
if B2 then
begin while B1 do S1 end
        else
begin while B1 do S2
end                                  (1)
```

[illegible]

```
while Bl do
begin if B2 then S1 else S2 end      (2)
```

[illegible] (1) [illegible] (2). [illegible] (1) [illegible] (2) [illegible]

[illegible]

[illegible] X[1:N] [illegible] Y [1:N] [illegible] (N=0) [illegible]

[illegible] j [illegible] j [illegible]:

```
j:=0;    [illegible]: = true;
while j≠N do
begin  j:=j+1;   [illegible]:=[illegible] □ (X[j] = Y[j])  end      (3)
```

[illegible]

```
j:=0;   [illegible] = true;
while j □ N^ [illegible] do
   begin j:=j+1; [illegible]:=(X[j] = Y[j]) end                    (4)
```

[illegible] (4) [illegible] (3) [illegible]

[illegible] X [illegible] Y [illegible] (N+1) [illegible] $_j$ [illegible] 0≤j≤N [illegible]

```
[illegible] j=0 PABH?_j=true
[illegible] j>0 [illegible]_j=[illegible]_j-1 ^ (X[j]=Y[j])        (5)
```

[illegible]

[illegible]=[illegible]$_N$

[illegible]

[illegible] := [illegible]$_j$ (6)

[illegible] j, [illegible] j=0.

[illegible] (3) [illegible] (4) [illegible] (7):

```
j:=0; [illegible] : = [illegible]_0;
while "[illegible]: [illegible] ≠ [illegible]_N"
do                                                            (7)
    begin j:=j+1;  [illegible]: =[illegible]_j end
```

[illegible] ≠ [illegible]$_N$", [illegible]

[illegible]=[illegible]$_j$

[illegible] (3) [illegible] (4) [illegible] "[illegible]" [illegible]$_N$.

[illegible] (3) [illegible]

$j=N$

[illegible]

[illegible]=[illegible]$_j$

[illegible] $j:=j+1$ [illegible]

[illegible] =[illegible]$_{j-1}$

[illegible]

[illegible]:=[illegible] ^ (X[j] - Y[j])

[illegible] (5).

[illegible] (4), [illegible] (5). [illegible] PABH?$_j$=**false**, [illegible] PABH?$_N$=**false** [illegible] PABH?$_j$=**false**, [illegible]$_j$=[illegible]$_N$. [illegible] "[illegible]=[illegible]$_N$," [illegible] (4). [illegible] (4), [illegible] "[illegible]=**true**", [illegible] (4) [illegible] "[illegible]: = [illegible]$_j$" [illegible]

[illegible]:=(X[j] =Y[j])

[illegible] (7) [illegible] (3) [illegible] (4) [illegible] ≠[illegible]$_N$" [illegible] (7) [illegible] (7) [illegible] (3), [illegible] (4), [illegible]

[illegible]

ᠬᠠᠪᠠᠷᠬᠠᠯᠠᠭᠠᠨᠤᠳ ᠪᠣᠯᠣᠨ ᠳᠤᠷᠤᠯᠳᠠᠭᠤᠨ ᠬᠣᠯᠪᠣᠭᠳᠣᠯ

ᠬᠣᠨᠣᠭᠳᠠᠭᠠᠨᠠ ᠬᠣᠨᠣᠭᠳᠠᠭᠠᠨᠠ ᠬᠣᠨᠣᠭᠳᠠᠭᠠᠨᠠ

Раздел 10. Дополнительные сведения

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Дополнительные сведения, которые должны быть приведены в отчете о результатах оценки регулирующего воздействия проекта акта, не предусмотренные разделами 1–9 настоящего примерного перечня, могут быть приведены по усмотрению разработчика.

Ѧ,ᶑ ɑɔɛʤɜɔɥ ɔ ɔ ɔɛɛɛ

12. [illegible]

[illegible]

[illegible]

[illegible]

```
begin
      . . . . . .
      . . . . . .
      . . . . . .
      . . . . . .
      . . . . . .
      . . . . . .
end
```

[illegible]

```
          begin
[illegible]:        . . . . . . . . . . .
[illegible]:        . . . . . . . . . . .
[illegible]:        . . . . . . . . . . .
          end
```

[illegible]

[illegible]:

```
          begin
[illegible]:      begin   . . . . . . .
                            . . . . . . . . end;
[illegible]:      begin   . . . . . . .
                            . . . . . . . . end;
[illegible]:      begin   . . . . . . .
                            . . . . . . . . end;
          end
```

[illegible] (begin-end) [illegible]

[illegible]

[illegible]; [illegible]; [illegible]

[illegible]

[illegible]

[illegible] EDSAC [*Wilkes [illegible]. V., Wheeler D. J., Gill S., The Preparation of Programsfor an Electronic Digital Computer; with Special Reference to the EDSAC and the use of a Library of Subroutins, Addison-Wesley Press, 1951.*] [illegible]

[illegible]

[illegible] -60, [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

(1) [illegible]

(2) [illegible]

(3) [illegible]

(4) [illegible]

(5) [illegible]

Algorithm Implementation/Sorting - 15

```
procedure Sort(a: array of Integer);
var
  tmp: Integer;
  i, j: Integer;
begin
  for i := 1 to Length(a) - 1 do
    begin
      j := i;
      while (j > 0) and (a[j - 1] > a[j]) do
        begin
          tmp := a[j];
          a[j] := a[j - 1];
          a[j - 1] := tmp;
          Dec(j);
        end;
    end;
end;
```

14. ד"ר ברוך בן יעקב, מומחה לרפואת הנשימה

ד"ר ברוך בן יעקב, מומחה לרפואת הנשימה, מקבל בקבלה פרטית בתל אביב ובמרכז רפואי שיבא. הוא מתמחה בטיפול בחולי אסתמה, בדלקת ריאות כronית ובמחלות ריאה חסימתיות. הוא מלווה חוקרים בתחום הרפואה הנשימתית ומפרסם מאמרים בכתבי עת בינלאומיים. בעבר כיהן כמנהל המחלקה הפנימית בבית חולים לב הדרום ובמרכז לרפואה דחופה בנגב. הוא בעל רישיון לרפואה פנימית ולרפואת ריאות, ומדריך רופאים בהתמחות. בזמנו הפרטי הוא אוהב לנגן בגיטרה ולטייל בשבילי הנגב.

15. מבוא לתכנות בשפת Pascal

בפרק זה נלמד מבוא לתכנות בשפת Pascal. נתחיל בתחביר הבסיסי, נעבור לסוגי נתונים ולמבני בקרה, ונסיים בכתיבת תוכנה קצרה. הדוגמאות בפרק מיועדות לקומפילציה בסביבת Turbo Pascal.

15.1 מבנה תוכנית

תוכנית Pascal מורכבת מכותרת, מבלוק הצהרות ומבלוק פקודות. הכותרת מתחילה במילת המפתח `program` ומסתיימת בנקודה. הבלוק מתחיל במילת המפתח `begin` ומסתיים ב`end` ואחריו נקודה.

15.2 הצהרות

1. הצהרות-קבוע מגדירות ערכים קבועים.
2. הצהרות-סוג מגדירות טיפוסים חדשים.
3. הצהרות-משתנים מגדירות משתנים.
4. הצהרות-תת-תוכנית מגדירות פונקציות ונהלים.

15.3 פקודות

פקודות מבוצעות בין `begin` ל`end`. פקודת הקצאה נכתבת `x := 5;`. פקודת `writeln` מדפיסה לפלט. פקודת `readln` קוראת מהקלט. פקודת `if ... then ... else` מבצעת בחירה. פקודת `while ... do` ו`for ... do` מבצעות לולאות.

15.4 דוגמה

```
program hello;
begin
  writeln('Hello, world');
end.
```

16. מבוא לתכנות בשפת Pascal

בפרק זה נלמד מבוא לתכנות בשפת Pascal. נתחיל בתחביר הבסיסי, נעבור לסוגי נתונים ולמבני בקרה, ונסיים בכתיבת תוכנה קצרה. הדוגמאות בפרק מיועדות לקומפילציה בסביבת Turbo Pascal.

16.1 מבנה תוכנית

תוכנית Pascal מורכבת מכותרת, מבלוק הצהרות ומבלוק פקודות. הכותרת מתחילה במילת המפתח `program` ומסתיימת בנקודה. הבלוק מתחיל במילת המפתח `begin` ומסתיים ב`end` ואחריו נקודה.

16.2 הצהרות

1. הצהרות-קבוע מגדירות ערכים קבועים.
2. הצהרות-סוג מגדירות טיפוסים חדשים.
3. הצהרות-משתנים מגדירות משתנים.
4. הצהרות-תת-תוכנית מגדירות פונקציות ונהלים.

16.3 פקודות

פקודות מבוצעות בין `begin` ל`end`. פקודת הקצאה נכתבת `x := 5;`. פקודת `writeln` מדפיסה לפלט. פקודת `readln` קוראת מהקלט. פקודת `if ... then ... else` מבצעת בחירה. פקודת `while ... do` ו`for ... do` מבצעות לולאות.

16.4 דוגמה

```
program hello;
begin
  writeln('Hello, world');
end.
```

15. [illegible]

[illegible]

[illegible]

[illegible] 100 [illegible]

```
0
01
010
0102
01020
010201
0102010
0102012
.......
```

[illegible] 1 [illegible] 100 [illegible] 101 [illegible]

```
 0
*00
 01
 010
*0100
*0101
 0102
 01020
*010200
 010201
 0102010
*01020100
*01020101
*01020102
*0102011
 0102012
 .......
```

[illegible]:

```
[illegible]
repeat if [illegible] then
    begin [illegible];
        [illegible] end
    else
        [illegible]
until [illegible]=101
```

[illegible] 100, [illegible] 101, [illegible] 100 [illegible] 101). [illegible]

[illegible]

```
[illegible];
repeat [illegible];
    while [illegible] do [illegible];
        [illegible]
until [illegible]=100
```

[illegible]:

```
[illegible];
repeat [illegible];
    [illegible]
until [illegible]=100
```

[illegible]

[illegible] s, [illegible] a^n+b^n [illegible]

($n=1$ $s=2=0^1-2^1=1^1+1^1$

$n=2$ $s=25=0^2=5^2=3^2+4^2$

$n=4$ $s=635318657=59^4-158^4=133^4+134^4$)

[illegible] a^n+b^n. [illegible] s [illegible]

[illegible]

```
integer s, k;
s:=l;
repeat s:=s+1;
    k:="[illegible] s [illegible] a^n+b^n"
until k>1
```

[illegible]:

```
integer k, s, t; t:=1 ([illegible]);
repeat s:="[illegible] t"
    k:="[illegible]"
    t:=s
until k>1
```

[illegible] s-[illegible] t, [illegible] t. [illegible] t [illegible]!

16. [illegible]

[illegible]

```
char x;
repeat [illegible] until x= [illegible]
```

[illegible]

1. [illegible]
2. [illegible]
3. [illegible] 0, 1, 2, 3, ... [illegible]

[illegible]

[illegible]

```
repeat [illegible] until [illegible];
```

[illegible]

1. [illegible]
2. [illegible]
3. [illegible]
4. [illegible]

[illegible]

```
repeat [illegible] until [illegible];
```

[illegible]

```
[illegible]:=true
```

[illegible]

```
[illegible]:=non [illegible]
```

[illegible]

```
boolean [illegible]; char x;
[illegible]:=true; [illegible];
repeat [illegible]
    [illegible]:=non [illegible]
until [illegible]
```

[illegible] 45 [illegible]

```
char array [illegible] [1:20]
```

[illegible]

```
[illegible][1]="[illegible]", [illegible][2]="[illegible]", [illegible][3]="[illegible]" [illegible] l=3.
```

[illegible]

```
l:=0;
repeat
l:=l+1; c[l]:=x; x:=[illegible] until x=[illegible]
```

[illegible]

```
while x=[illegible] do x:=[illegible]
```

[illegible] (33). [illegible]

```
if [illegible] then begin k:=0; [illegible]:=+1; [illegible]:=l end
            else begin k:=l+1; [illegible]:=-1; [illegible]:=1 end;
repeat k:=k+[illegible]; [illegible]([illegible][k]) until k=[illegible]
```

[illegible]

```
if x=[illegible] then [illegible]([illegible]) else [illegible]([illegible])
```

[illegible]

```
boolean [illegible]; char x; char array [illegible][1:20]; integer l, k, [illegible], [illegible];
[illegible]:=true; x:=[illegible];
repeat l:=0;
    repeat l:=l+1; c[l]:=x; x:=[illegible] until x=[illegible];
    while x=[illegible] do [illegible]:=[illegible];
    if [illegible] then begin k:=0; [illegible]:=+1; [illegible]:=l end
        else begin k:=l+1; [illegible]:=-1; [illegible]:=1 end
    repeat k:=k+[illegible]; [illegible]([illegible][k]) until k=[illegible];
    if x=[illegible] then [illegible]([illegible]) else [illegible]([illegible]);
    [illegible]:=non [illegible]
until x=[illegible]
```

[illegible]

Alphanumeric Sorting Implementation in Pascal

Alphanumeric sorting is a common requirement when dealing with data that contains both letters and numbers, such as file names or product codes. Standard lexicographical sorting treats numbers as characters, leading to results like "1", "10", "2" instead of the expected "1", "2", "10". This implementation provides a custom sorting algorithm in Pascal that correctly handles alphanumeric strings by treating numbers as numerical values.

```
procedure AlphaNumSort(var A: array[1..N] of string);
var
  i, j, k, N: integer;
  temp: string;
  isNum: boolean;
  numA, numB: integer;
begin
  N := Length(A);
  for i := 1 to N do
  begin
    for j := 1 to N - i do
    begin
      if A[j] > A[j+1] then
      begin
        temp := A[j];
        A[j] := A[j+1];
        A[j+1] := temp;
      end;
    end;
  end;
end;
```

```
procedure AlphaNumSort(var A: array[1..N] of string);
var
  i, j, k, N: integer;
  temp: string;
  isNum: boolean;
  numA, numB: integer;
begin
  N := Length(A);
  for i := 1 to N do
  begin
    for j := 1 to N - i do
    begin
      if A[j] > A[j+1] then
      begin
        temp := A[j];
        A[j] := A[j+1];
        A[j+1] := temp;
      end;
    end;
  end;
end
```