Informatyka 1

Wykład IV

Wyrażenia i instrukcje cd., ręczna symulacja, operacje wejścia/wyjścia

Robert Muszyński
ZPCiR IIAiR PWr

Zagadnienia: instrukcja warunkowa `CASE-OF-END`, instrukcja pętli `REPEAT-UNTIL`, ręczna symulacja, operacje wejścia/wyjścia, wbudowane procedury i funkcje w Pascalu.

– Skład FoilTEX –

Instrukcja warunkowa CASE-OF-END

```
instrukcja-CASE = "CASE" wyrazenie "OF"
                  element-listy-wyboru { ";"
                  element-listy-wyboru } [ ";" ] "END".
```

Instrukcja warunkowa CASE-OF-END

```
instrukcja-CASE = "CASE" wyrazenie "OF"
                  element-listy-wyboru { ";"
                  element-listy-wyboru } [ ";" ] "END".
element-listy-wyboru = lista-stalych ":" instrukcja.
```

Instrukcja warunkowa CASE-OF-END

```
instrukcja-CASE = "CASE" wyrazenie "OF"
                  element-listy-wyboru { ";"
                  element-listy-wyboru } [ ";" ] "END".
element-listy-wyboru = lista-stalych ":" instrukcja.
```

Przykład: dodawanie lub mnożenie

```
CASE Oper OF
```

Instrukcja warunkowa CASE-OF-END

```
instrukcja-CASE = "CASE" wyrazenie "OF"
                  element-listy-wyboru { ";"
                  element-listy-wyboru } [ ";" ] "END".
element-listy-wyboru = lista-stalych ":" instrukcja.
```

Przykład: dodawanie lub mnożenie

```
CASE Oper OF
   '+': Z := X + Y;
```

Instrukcja warunkowa CASE-OF-END

```
instrukcja-CASE = "CASE" wyrazenie "OF"
                  element-listy-wyboru { ";"
                  element-listy-wyboru } [ ";" ] "END".
element-listy-wyboru = lista-stalych ":" instrukcja.
```

Przykład: dodawanie lub mnożenie

```
CASE Oper OF
   '+': Z := X + Y;
   '*': Z := X * Y;
```

Instrukcja warunkowa CASE-OF-END

```
instrukcja-CASE = "CASE" wyrazenie "OF"
                  element-listy-wyboru { ";"
                  element-listy-wyboru } [ ";" ] "END".
element-listy-wyboru = lista-stalych ":" instrukcja.
```

Przykład: dodawanie lub mnożenie

```
CASE Oper OF
  '+': Z := X + Y;
  '*': Z := X * Y;
  '-','/': WRITELN('Operacja niedozwolona: "',
                   Oper,'".')
```

Instrukcja warunkowa CASE-OF-END

```
instrukcja-CASE = "CASE" wyrazenie "OF"
                  element-listy-wyboru { ";"
                  element-listy-wyboru } [ ";" ] "END".
element-listy-wyboru = lista-stalych ":" instrukcja.
```

Przykład: dodawanie lub mnożenie

```
CASE Oper OF
  '+': Z := X + Y;
  '*': Z := X * Y;
  '-','/': WRITELN('Operacja niedozwolona: "',
                   Oper,'".')
END
```

Instrukcja pętli REPEAT-UNTIL

```
instrukcja-REPEAT = "REPEAT" ciag-instrukcji
                    "UNTIL" wyrazenie-logiczne.
```

Instrukcja pętli REPEAT-UNTIL

```
instrukcja-REPEAT = "REPEAT" ciag-instrukcji
                    "UNTIL" wyrazenie-logiczne.
ciag-instrukcji = instrukcja { ";" ciag-instrukcji }.
```

Instrukcja pętli REPEAT-UNTIL

```
instrukcja-REPEAT = "REPEAT" ciag-instrukcji
                    "UNTIL" wyrazenie-logiczne.
ciag-instrukcji = instrukcja { ";" ciag-instrukcji }.
wyrazenie-logiczne = wyrazenie.
```

Instrukcja pętli REPEAT-UNTIL

```
instrukcja-REPEAT = "REPEAT" ciag-instrukcji
                    "UNTIL" wyrazenie-logiczne.
ciag-instrukcji = instrukcja { ";" ciag-instrukcji }.
wyrazenie-logiczne = wyrazenie.
```

Przykład: algorytm Euklidesa — wej.: `n`, `m`; wyj.: `n`

```
REPEAT
  k := n MOD m;
  n := m;
  m := k;
UNTIL m = 0;
```

Ręczna symulacja

Przykład: zamiana wartości zmiennych A i B

```
A := 5;
B := 10;

A := B;
B := A;
```


Ręczna symulacja

Przykład: zamiana wartości zmiennych A i B

```
1   A := 5;
2   B := 10;

3   A := B;
4   B := A;
```


Ręczna symulacja

Przykład: zamiana wartości zmiennych A i B

```
1   A := 5;
2   B := 10;

3   A := B;
4   B := A;
```

nr linii	A	B	...	Uwagi
	?	?		

Ręczna symulacja

Przykład: zamiana wartości zmiennych A i B

```
1   A := 5;
2   B := 10;

3   A := B;
4   B := A;
```

nr linii	A	B	...	Uwagi
	?	?		
1	5			

Ręczna symulacja

Przykład: zamiana wartości zmiennych A i B

```
1   A := 5;
2   B := 10;

3   A := B;
4   B := A;
```

nr linii	A	B	...	Uwagi
	?	?		
1	5			
2		10		

Ręczna symulacja

Przykład: zamiana wartości zmiennych A i B

```
1   A := 5;
2   B := 10;

3   A := B;
4   B := A;
```

nr linii	A	B	...	Uwagi
	?	?		
1	5			
2		10		
3	10			

Ręczna symulacja

Przykład: zamiana wartości zmiennych A i B

```
1   A := 5;
2   B := 10;

3   A := B;
4   B := A;
```

nr linii	A	B	...	Uwagi
	?	?		
1	5			
2		10		
3	10			
4		10		

```
N := 27;
Pierwsza := TRUE;
Dzielnik := 2;
  WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
  IF (N MOD Dzielnik) = 0
    THEN Pierwsza := FALSE
    ELSE Dzielnik := Dzielnik + 1;
...
```

```
1  N := 27;
2  Pierwsza := TRUE;
3  Dzielnik := 2;
4    WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5    IF (N MOD Dzielnik) = 0
6      THEN Pierwsza := FALSE
7      ELSE Dzielnik := Dzielnik + 1;
8  ...
```

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	
4				warunek `WHILE`: `TRUE`

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `FALSE`

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `FALSE`
7			3	

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `FALSE`
7			3	
4				warunek `WHILE`: `TRUE`

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `FALSE`
7			3	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `TRUE`

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `FALSE`
7			3	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `TRUE`
6		FALSE		

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `FALSE`
7			3	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `TRUE`
6		FALSE		
4				warunek `WHILE`: `FALSE`

```
1   N := 27;
2   Pierwsza := TRUE;
3   Dzielnik := 2;
4     WHILE Pierwsza AND (Dzielnik <= N DIV 2) DO
5     IF (N MOD Dzielnik) = 0
6       THEN Pierwsza := FALSE
7       ELSE Dzielnik := Dzielnik + 1;
8   ...
```

nr linii	N	Pierwsza	Dzielnik	Uwagi
	?	?	?	
1	27			
2		TRUE		
3			2	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `FALSE`
7			3	
4				warunek `WHILE`: `TRUE`
5				warunek `IF`: `TRUE`
6		FALSE		
4				warunek `WHILE`: `FALSE`
8				

Operacje wejścia/wyjścia

- instrukcje `READ/READLN` oraz `WRITE/WRITELN` operują na strumieniach danych;

Operacje wejścia/wyjścia

- instrukcje `READ/READLN` oraz `WRITE/WRITELN` operują na strumieniach danych;
- dwa standardowe strumienie danych są zawsze dostępne, jednak program musi zadeklarować ich użycie:

```
PROGRAM Testy(INPUT,OUTPUT);
```

Operacje wejścia/wyjścia

- instrukcje `READ/READLN` oraz `WRITE/WRITELN` operują na strumieniach danych;
- dwa standardowe strumienie danych są zawsze dostępne, jednak program musi zadeklarować ich użycie:

```
PROGRAM Testy(INPUT,OUTPUT);
```

- standardowe strumienie danych zwykle utożsamiane są z klawiaturą i ekranem terminala, jednakże system Unix pozwala przypisać dowolny z nich do plików dyskowych

```
%testy <testy.dane >testy.wyniki
```

Operacje wejścia/wyjścia

- instrukcje `READ/READLN` oraz `WRITE/WRITELN` operują na strumieniach danych;
- dwa standardowe strumienie danych są zawsze dostępne, jednak program musi zadeklarować ich użycie:

```
PROGRAM Testy(INPUT,OUTPUT);
```

- standardowe strumienie danych zwykle utożsamiane są z klawiaturą i ekranem terminala, jednakże system Unix pozwala przypisać dowolny z nich do plików dyskowych

```
%testy <testy.dane >testy.wyniki
```

- mechanizm skierowania strumieni można wykorzystać przy testowaniu programów.

Instrukcje READ/READLN

```
READ(v1,...,vn)
READLN(v1,...,vn)
```

`vi` — zmienna

Instrukcje READ/READLN

```
READ(v1,...,vn)
READLN(v1,...,vn)
```

`vi` — zmienna

Przykłady:

```
READ(oper);
```

Instrukcje READ/READLN

```
READ(v1,...,vn)
READLN(v1,...,vn)
```

vi — zmienna

Przykłady:

```
READ(oper);
READLN(arg1,arg2);
```

Instrukcje READ/READLN

```
READ(v1,...,vn)
READLN(v1,...,vn)
```

`vi` — zmienna

Przykłady:

```
READ(oper);
READLN(arg1,arg2);
READLN;
```

Instrukcje WRITE/WRITELN

```
WRITE(e1,...,en)
WRITELN(e1,...,en)
```

`ei` — wyrażenie lub zmienna

Instrukcje WRITE/WRITELN

```
WRITE(e1,...,en)
WRITELN(e1,...,en)
```

ei — wyrażenie lub zmienna

Przykłady:

```
WRITELN('Oto jestem, Swiecie');
```

Instrukcje WRITE/WRITELN

```
WRITE(e1,...,en)
WRITELN(e1,...,en)
```

ei — wyrażenie lub zmienna

Przykłady:

```
WRITELN('Oto jestem, Swiecie');
{ mieszane dane napisowe i liczby INTEGER }
WRITELN('  a= ', a:8, '  b= ', b:8);
```

Instrukcje WRITE/WRITELN

```
WRITE(e1,...,en)
WRITELN(e1,...,en)
```

ei — wyrażenie lub zmienna

Przykłady:

```
WRITELN('Oto jestem, Swiecie');
{ mieszane dane napisowe i liczby INTEGER }
WRITELN('  a= ', a:8, '  b= ', b:8);
{ wynik typu BOOLEAN }
WRITE('Wyroznik nieujemny? ', b*b-4*a*c>=0.0);
```

Instrukcje WRITE/WRITELN

```
WRITE(e1,...,en)
WRITELN(e1,...,en)
```

ei — wyrażenie lub zmienna

Przykłady:

```
WRITELN('Oto jestem, Swiecie');
{ mieszane dane napisowe i liczby INTEGER }
WRITELN('  a= ', a:8, '  b= ', b:8);
{ wynik typu BOOLEAN }
WRITE('Wyroznik nieujemny? ', b*b-4*a*c>=0.0);
{ liczby REAL }
WRITE('Wyniki: x1=',x2:3,', oraz x2=',x2:0:0);
```

Operacje wejścia/wyjścia na plikach

Strukutra programu:

```
PROGRAM prog1(INPUT,OUTPUT,plik1,plik2);

VAR plik1,plik2: TEXT;
```

Operacje wejścia/wyjścia na plikach

Strukutra programu:

```
PROGRAM prog1(INPUT,OUTPUT,plik1,plik2);

VAR plik1,plik2: TEXT;

{... przyklad wywolania:}
READLN(plik1,Oper,Arg1,Arg2);
```

Operacje wejścia/wyjścia na plikach

Strukutra programu:

```
PROGRAM prog1(INPUT,OUTPUT,plik1,plik2);

VAR plik1,plik2: TEXT;

{... przyklad wywolania:}
READLN(plik1,Oper,Arg1,Arg2);
{...}
WRITE(plik2,'Wartosc koncowa: ',Arg1);
```


Operacje wejścia/wyjścia na plikach

Inicjalizacja plików:

```
RESET(plik1,'arytmetyka.d');
```

Operacje wejścia/wyjścia na plikach

Inicjalizacja plików:

```
RESET(plik1,'arytmetyka.d');
REWRITE(plik2,'wyniki.txt');
```


Operacje wejścia/wyjścia na plikach

Inicjalizacja plików:

```
RESET(plik1,'arytmetyka.d');
REWRITE(plik2,'wyniki.txt');
APPEND(plik2,'wyniki.txt');
```

Operacje wejścia/wyjścia na plikach

Inicjalizacja plików:

```
RESET(plik1,'arytmetyka.d');
REWRITE(plik2,'wyniki.txt');
APPEND(plik2,'wyniki.txt');
CLOSE(plik1);
```

Operacje wejścia/wyjścia na plikach

Inicjalizacja plików:

```
RESET(plik1,'arytmetyka.d');
REWRITE(plik2,'wyniki.txt');
APPEND(plik2,'wyniki.txt');
CLOSE(plik1);
```

Przydatne funkcje:

```
EOLN(tekst1);
```

Operacje wejścia/wyjścia na plikach

Inicjalizacja plików:

```
RESET(plik1,'arytmetyka.d');
REWRITE(plik2,'wyniki.txt');
APPEND(plik2,'wyniki.txt');
CLOSE(plik1);
```

Przydatne funkcje:

```
EOLN(tekst1);
EOF(tekst1);
```

Wbudowane procedury i funkcje w Pascalu

Różnice między procedurami a funkcjami:

- funkcja wylicza i zwraca wartość, a procedura nie;

Wbudowane procedury i funkcje w Pascalu

Różnice między procedurami a funkcjami:

- funkcja wylicza i zwraca wartość, a procedura nie;
- procedura stanowi instrukcję i może wystąpić w programie jedynie w sekwencji instrukcji;

Wbudowane procedury i funkcje w Pascalu

Różnice między procedurami a funkcjami:

- funkcja wylicza i zwraca wartość, a procedura nie;
- procedura stanowi instrukcję i może wystąpić w programie jedynie w sekwencji instrukcji;
- funkcja stanowi wyrażenie i może wystąpić w programie jedynie w roli wyrażenia.

Wbudowane procedury i funkcje w Pascalu

Przykłady procedur:

- `READ/READLN,`
- `WRITE/WRITELN,`
- `RESET/REWRITE, CLOSE,`
- `HALT;`

Wbudowane procedury i funkcje w Pascalu

Przykłady procedur:

- READ/READLN,
- WRITE/WRITELN,
- RESET/REWRITE, CLOSE,
- HALT;

Przykłady funkcji:

- SQRT, SQR,

Wbudowane procedury i funkcje w Pascalu

Przykłady procedur:

- READ/READLN,
- WRITE/WRITELN,
- RESET/REWRITE, CLOSE,
- HALT;

Przykłady funkcji:

- SQRT, SQR,
- ABS,
- SIN, COS,

Wbudowane procedury i funkcje w Pascalu

Przykłady procedur:

- READ/READLN,
- WRITE/WRITELN,
- RESET/REWRITE, CLOSE,
- HALT;

Przykłady funkcji:

- SQRT, SQR,
- ABS,
- SIN, COS,
- EXP, LN,

Wbudowane procedury i funkcje w Pascalu

Przykłady procedur:

- READ/READLN,
- WRITE/WRITELN,
- RESET/REWRITE, CLOSE,
- HALT;

Przykłady funkcji:

- SQRT, SQR,
- ABS,
- SIN, COS,
- EXP, LN,
- ROUND, TRUNC,

Wbudowane procedury i funkcje w Pascalu

Przykłady procedur:

- READ/READLN,
- WRITE/WRITELN,
- RESET/REWRITE, CLOSE,
- HALT;

Przykłady funkcji:

- SQRT, SQR,
- ABS,
- SIN, COS,
- EXP, LN,
- ROUND, TRUNC,
- ODD,

Wbudowane procedury i funkcje w Pascalu

Przykłady procedur:

- READ/READLN,
- WRITE/WRITELN,
- RESET/REWRITE, CLOSE,
- HALT;

Przykłady funkcji:

- SQRT, SQR,
- ABS,
- SIN, COS,
- EXP, LN,
- ROUND, TRUNC,
- ODD,
- EOLN, EOF

oraz

- `SUCC(w)` — wylicza następną po `w` wartość w typie porządkowym,

- `SUCC(w)` — wylicza następną po `w` wartość w typie porządkowym,
- `PRED(w)` — wylicza poprzednią przed `w` wartość w typie porządkowym,

- `SUCC(w)` — wylicza następną po `w` wartość w typie porządkowym,
- `PRED(w)` — wylicza poprzednią przed `w` wartość w typie porządkowym,
- `ORD(w)` — wylicza numer wartości `w` w typie porządkowym,

- `SUCC(w)` — wylicza następną po `w` wartość w typie porządkowym,
- `PRED(w)` — wylicza poprzednią przed `w` wartość w typie porządkowym,
- `ORD(w)` — wylicza numer wartości `w` w typie porządkowym,
- `CHR(n)` — wylicza znak o numerze porządkowym `n`.

- `SUCC(w)` — wylicza następną po `w` wartość w typie porządkowym,
- `PRED(w)` — wylicza poprzednią przed `w` wartość w typie porządkowym,
- `ORD(w)` — wylicza numer wartości `w` w typie porządkowym,
- `CHR(n)` — wylicza znak o numerze porządkowym `n`.

```
SUCC(PRED(w)) = w
PRED(SUCC(w)) = w
```

- `SUCC(w)` — wylicza następną po `w` wartość w typie porządkowym,
- `PRED(w)` — wylicza poprzednią przed `w` wartość w typie porządkowym,
- `ORD(w)` — wylicza numer wartości `w` w typie porządkowym,
- `CHR(n)` — wylicza znak o numerze porządkowym `n`.

```
SUCC(PRED(w)) = w  |  CHR(ORD(w)) = w
PRED(SUCC(w)) = w  |  ORD(CHR(n)) = n
```

Indeks

- Instrukcja warunkowa `CASE-OF-END`
- Instrukcja pętli `REPEAT-UNTIL`
- Ręczna symulacja
- Operacje wejścia/wyjścia
- Instrukcje `READ/READLN`
- Instrukcje `WRITE/WRITELN`
- Operacje wejścia/wyjścia na plikach
- Wbudowane procedury i funkcje w Pascalu