Programowanie modułowe

Moduł jest konstrukcją języka Pascal umożliwiającą grupowanie deklaracji typów, stałych, zmiennych i procedur w oddzielnych plikach. Koncepcja modułu pozwala na podział tekstu całego programu na części przechowywane w oddzielnych plikach. Wyodrębnione części programu mogą być osobno opracowywane i kompilowane.

Struktura modułu w Pascalu

Struktura modułu jest podobna do struktury programu. Moduł składa się z nagłówka, części deklaracyjnej i części wykonawczej. W odróżnieniu od programu część deklaracyjna modułu dzieli się na część publiczną zawierającą deklaracje obiektów, które powinny być znane poza modułem i część prywatną zawierającą deklaracje obiektów lokalnych. Część publiczna rozpoczyna się słowem **interface** i poprzedza część publiczną. Część prywatna rozpoczyna się słowem **implementation**.

Budowa modułu:

unit <nazwa>;
interface
<deklaracje - część publiczna>
implementation
<deklaracje - część prywatna>
begin
<część wykonawcza - inicjująca>
end.

Deklaracje procedur występujące w części publicznej mają postać zapowiedzi i powinny zawierać tylko nagłówki.
Część prywatna powinna zawierać deklaracje procedur lokalnych oraz kompletne deklaracje procedur publicznych, których nagłówki umieszczono w części interfejsowej.
Część wykonawcza modułu jest przeznaczona na umieszczenie w niej instrukcji inicjujących moduł i często pozostawiana jest pusta. Jeżeli część wykonawcza nie zawiera żadnych instrukcji, a więc składa się jedynie ze słów **begin** i **end** , to słowo kluczowe **begin** może zostać pominięte.

Dyrektywa Uses

Deklaracje zawarte w części publicznej modułu stają się widoczne w programie w wyniku umieszczenia w nim dyrektywy **uses** o postaci:

uses <lista nazw modułów>

Dyrektywa ta musi występować w części deklaracyjnej programu jako pierwsza.
W czasie tłumaczenia programu kompilator dołącza wszystkie obiekty modułów z listy dyrektywy **uses**.
Uruchomienie skompilowanego programu powoduje kolejne wykonanie części inicjujących wszystkich modułów z listy, a następnie części wykonawczej programu głównego.

Biblioteki

Moduły stosuje się do gromadzenia w jedna całość pewnych zestawów procedur (wraz z towarzyszącymi strukturami danych). Tak zgrupowane zestawy procedur nazywane są bibliotekami.

Przykład

```
unit Ciagi; { moduł – biblioteka przetwarzania ciągów – pliku CIAGI.PAS}
interface

const max_wym = 10;
type Tab = array [1..max_wym] of real;

procedure czytaj(n: integer; var x: tab);
function Suma(n: integer; x: tab): real;
function Srednia (n: integer; x: tab): real;

implementation

function Suma(n: integer; x: tab): real;
var i: integer;
    s: real;
begin
s:=0;
for i:=1 to n do s:=s+x[i];
suma:=s;
end;

function Srednia (n: integer; x: tab): real;
begin
if n>0 then srednia:=suma(n, x)/n else srednia:=0;
end;

procedure czytaj(n: integer; var x: tab);
var i:integer;
begin
writeln('Podaj elementy ciagu');
for i:=1 to n do read(x[i]);
end;

end.

program prog1;
{ Program wykorzystujący bibliotekę Ciagi pamiętany w pliku PROG1.DPR }
uses Ciagi;

var x: tab;
     n: integer;
    m1, m2: real;
```

```
begin
Writeln('Podaj liczność ciągu');
Readln(n);
Czytaj(n, x);
Writeln('Średnia=', srednia(n, x):10:2);
end.
```

Zapis modułu powinien być umieszczony w pliku o nazwie tożsamej z nazwą modułu wymienioną w nagłówku i rozszerzeniu PAS. W przypadku modułu **Ciagi** zawartego w przykładzie nazwa pliku tekstowego zawierającego moduł powinna być CIAGI.PAS.
Jeżeli nazwa modułu jest dłuższa niż 8 znaków to nazwa pliku powinna być identyczna z pierwszymi 8 znakami.
Istnieje możliwość nadania dowolnej nazwy pliku zawierającego moduł poprzez wykorzystanie dyrektywy kompilatora {$U nazwa}. Dyrektywa ta zawierająca nazwę pliku powinna być umieszczona w nagłówku modułu bezpośrednio za słowem **unit**, a przed nazwą modułu.

Kompilacja modułów

Translacja programu składającego się z pliku programu głównego i plików modułu odbywa się w dwóch fazach.

Dla programu z przytoczonego powyżej przykładu pierwszą fazę można schematycznie przedstawić następująco:

CIAGI.PAS $\Rightarrow$ CIAGI.DCU

W fazie tej odbywa się kompilacja modułu i utworzenie pliku binarnego zawierającego skompilowany moduł
W drugiej fazie odbywa się kompilacja programu głównego, konsolidacja programu i utworzenie binarnego pliku wykonywalnego.

CIAGI.DCU + PROG1.DPR $\Rightarrow$ PROG1.EXE

Obie fazy mogą być oddzielone w czasie. Ponieważ do utworzenia programu wykonywalnego niepotrzebny jest plik źródłowy modułu, to utworzone wcześniej biblioteki mogą być rozpowszechniane w postaci skompilowanych plików *.DCU.

Moduły zagnieżdżone

Deklaracje **uses** mogą występować w modułach tzn. moduły mogą wykorzystywać inne moduły.
Dyrektywa **uses** może być umieszczana w module w dwóch miejscach:

- bezpośrednio za słowem kluczowym **implementation** – jeżeli zasoby deklarowanego modułu są wykorzystywana jedynie w części prywatnej,
- bezpośrednio za słowem kluczowym **interface**, jeśli zasoby te są wykorzystywane także w części publicznej.

Dla poprawnej kompilacji modułu zawierającego dyrektywę **uses** konieczna jest dostępność plików *.DCU dla wszystkich modułów wymienionych w tej dyrektywie. Oznacza to, że kolejność kompilacji modułów dużego projektu jest wymuszana poprzez zależności pomiędzy nimi – jako pierwsze powinny być kompilowane te moduły, które nie są zależne od innych.

Dopuszczalne jest wzajemne wykorzystywanie zasobów dwu modułów jak w przykładzie zamieszczonym poniżej:

unit pierwszy;
interface
uses drugi;
{ deklaracje – część publiczna }
implementation
{ deklaracje – część prywatna }
end.

unit drugi;
interface
{ deklaracje – część publiczna }
implementation
uses pierwszy;
{ deklaracje – część prywatna }
end.

W przypadku takiego wzajemnego wykorzystywania zasobów modułów obowiązują dwa ograniczenia:

- moduły wzajemnie zależne powinny być kompilowane jednocześnie (łącznie z programem je wykorzystującym),
- przynajmniej w jednym z dwu modułów wzajemnie zależnych dyrektywa **uses** zawierająca wyszczególnienie drugiego modułu powinna występować w części prywatnej. Sytuacja taka występuje w podanym wyżej przykładzie.

Moduły standardowe

W Turbo Pascalu użytkownik ma możliwość korzystania z kilku modułów standardowych, które są dostarczane w pakiecie języka. Najważniejsze z nich to:

System - biblioteka procedur standardowych,
Crt - biblioteka procedur do organizacji współpracy z ekranem i klawiaturą;
Printer - biblioteka procedur zarządzających drukarką,
Dos - biblioteka procedur umożliwiająca korzystanie z zaawansowanych usług Dosu,
Graph - biblioteka graficzna.

W przypadku języka Delphi ilość bibliotek standardowych dostarczanych w pakiecie jest bardzo duża.