

LISTY PODWIESZANE

1. Z listą podwieszoną mamy do czynienia, gdy lista (tzw. lista główna) zawiera wskaźniki do innej listy bądź list – niejako podwieszonych do listy głównej.

2. **Przykład:**

Napisz uproszczony program do obsługi biblioteki, który zawiera:

- Procedurę dodawania nowych książek (w sposób posortowany po numerze)
- Procedurę tworzenia nowych użytkowników (studentów)
- Procedurę do wypożyczania książek przez studentów
- Procedurę wyświetlającą wszystkie książki w bibliotece
- Procedurę wyświetlającą wszystkich studentów wraz z informacją o wypożyczonych książkach (lub informacją, że żadnej nie wypożyczył).

- a. Moduł do obsługi biblioteki

```
unit biblioteka;
interface
type
pKsiazka = ^Tksiazka;
pstudent = ^Tstudent;

Tksiazka = record
    id:Integer;
    tytul, autor: String[20];
    next: pKsiazka;
end;

Tstudent = record
    nazwisko:string[30];
    ksiazki: pksiazka;
    next: pstudent;
end;

// procedura dodajaca studenta na poczatke Listy studentow
procedure dodajS(var listaS:pStudent; nazwisko:string);

// procedura dodajaca nowa ksiazke do listy ksiazek sortujaco po ID
procedure dodajK(var listaK:pksiazka; id:Integer; autor,tytul:String);

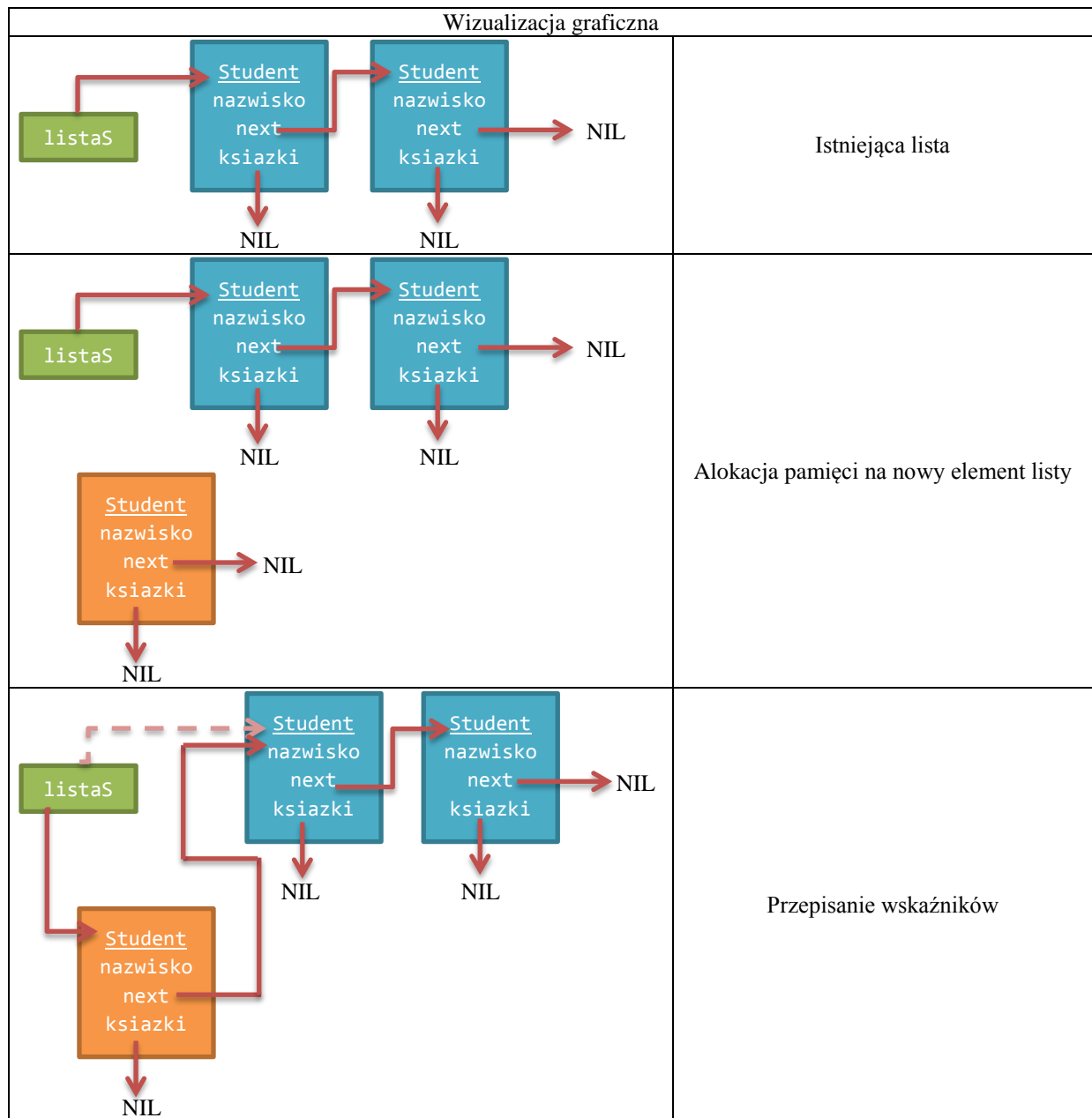
// procedura wyswietlajaca wszystkie ksiazki
procedure printK(tmp: pKsiazka);

// procedura przypisujaca wybrana ksiazke do listy wypożyczonych ksiazek
// wybranego studenta
procedure wypożycz(listak:pksiazka; var listas:pstudent; komu:string;
co:string);

// procedura wyswietlajaca wszystkich studentow wraz z informacja o
// wypożyczonych ksiazkach
procedure wyswietlS(listaS:pstudent);
```

implementation

```
procedure dodajS(var listaS:pStudent; nazwisko:string);  
var  
    nowy: pstudent;  
begin  
    new( nowy );  
    nowy^.nazwisko := nazwisko;  
    nowy^.ksiazki:=NIL;  
    nowy^.next := listaS;  
    listaS := nowy;  
end;
```

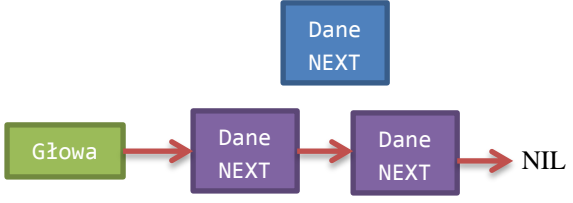
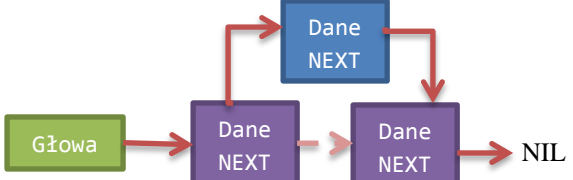


```

procedure dodajK(var listaK:pksiazka; id:Integer; autor,tytul:String);
var
    nowy,tmp: pKsiazka;
begin
    // alokacja pamięci na nowy element listy
    new( nowy );
    // ustawienie wartości nowego elementu
    nowy^.id := id;
    nowy^.autor := autor;
    nowy^.tytul := tytul;
    nowy^.next := NIL;

    // jeżeli lista nie jest pusta
    // oraz numer ID pierwszego elementu jest mniejszy od zadanego id
    if (listak <> NIL) AND (listak^.id < id) then
    begin
        // iterujemy po liście tak długo aż:
        // dojdziemy do ostatniego elementu listy lub
        // id następnika następnego elementu będzie mniejsze od zadanego id
        tmp:= listak;
        while (tmp ^.next <> NIL) AND (tmp ^.next^.id < id) do
        begin
            tmp:= tmp ^.next;
        end;
        // wstawiamy w wybranym miejscu i przepisujemy wskaźniki
        nowy^.next := tmp ^.next;
        tmp ^.next := nowy;
    end
    else
    begin
        // w przeciwnym wypadku - dodajemy na początek listy
        nowy^.next := listaK;
        listaK := nowy;
    end;
end;

```

Wizualizacja graficzna	
	<p>Jeżeli wartość nowo utworzonego elementu jest mniejsza od pierwszego elementu – postępujemy analogicznie jak w procedurze pop.</p> <p>W przeciwnym wypadku – znajdujemy element którego wartość następnika jest większa od wartości nowego elementu (na rysunku jasno niebieski)</p>
	<p>Przepisanie wskaźników</p>

```

procedure printK(tmp: pKsiazka);
begin
    while( tmp <> NIL )do
    begin
        Writeln( tmp^.id:4,tmp^.autor,' ',tmp^.tytul);
        tmp := tmp ^.next;
    end;
end;

procedure wypożycz(listak:pksiazka; var listas:pstudent; komu:string;
                    tytul:string);
var
    pointerK:pKsiazka;
    pointerS:pstudent;
begin
    // przypisanie wskaźników na początki obu list
    pointerK:=listak;
    pointerS:=listas;

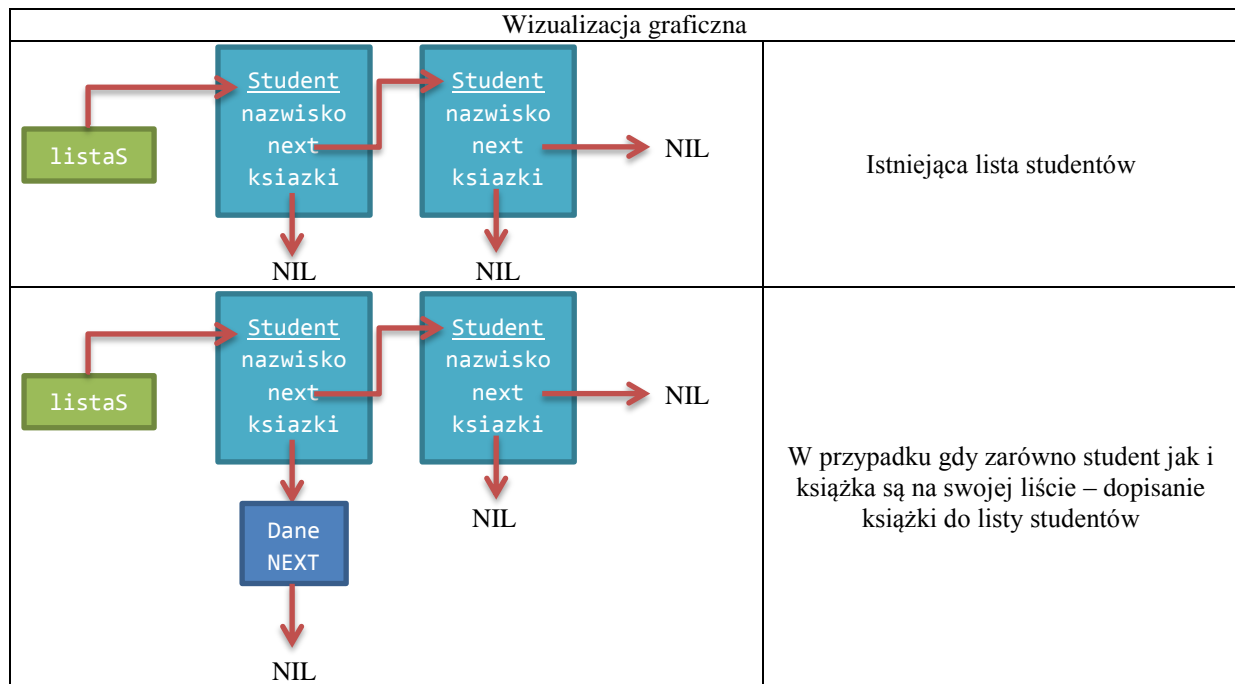
    // iteracja po liście studentów w poszukiwaniu zadanego studenta
    while(pointerS <> nil) do
    begin
        // jeśli student został odnaleziony na liście - poleceniem break
        // wychodzimy z pętli while
        if(pointerS^.nazwisko = komu) then
            break
        else
            pointerS:=pointerS^.next;
    end;

    // jeżeli po wyjściu z pętli while, wskaźnik wskazuje na nil oznacza to
    // że zadanego studenta nie ma na liście - wyświetlamy komunikat
    if(pointerS = nil) then
        writeln('Brak studenta');

    // analogicznie jak w przypadku studentów, sprawdzamy czy zadana książka jest
    // na liście książek
    while(pointerK <> nil) do
    begin
        if(pointerK^.tytul = tytul) then
            break
        else
            pointerK:=pointerK^.next;
    end;
    if(pointerK = nil) then
        writeln('Brak książki');

    // jeżeli zarówno student jak i książka znajdują się na swoich listach
    // za pomocą procedury dodajK dodajemy wybraną książkę do listy książek
    // wybranego studenta
    if((pointerS <> nil)and(pointerK <> nil)) then
        dodajK(pointerS^.ksiazki,PointerK^.id, PointerK^.autor, PointerK^.tytul)
    else
        // w przeciwnym razie wyświetlamy komunikat, że operacja się nie powiodła
        writeln('nie dodano');
    end;
end;

```



```

procedure wyswietlS(listaS:pstudent);
var
  ksiazki:pKsiazka;
begin
  // iterujemy po liście studentów
  while( listaS <> NIL )do
    begin
      writeln('-----');
      Writeln( listaS^.nazwisko);
      // sprawdzamy czy student wypożyczył jakieś książki
      if(listaS^.ksiazki = NIL) then
        // jeśli nie - wyświetlamy odpowiedni komunikat
        writeln('Student nie czyta')
      else
        begin
          // w przeciwnym wypadku iterujemy po podwieszanej liście książek
          // i wyświetlamy jej zawartość
          while(listaS^.ksiazki<>NIL) do
            begin
              Writeln( listaS^.ksiazki^.id:4,listaS^.ksiazki^.autor,'
',listaS^.ksiazki^.tytul);
              listaS^.ksiazki:=listaS^.ksiazki^.next;
            end;
          end;
          listaS := listaS ^.next;
        end;
      end;
    end;
end;

```

b. Program główny

```
program project1;

uses Unit1;

var
  ksiazki:pKsiazka=NIL;
  studenci:pstudent=NIL;                                // 0

begin
  dodajK(ksiazki,0,' Joudi','Bez mojej zgody');          // 1
  dodajK(ksiazki,2,' Tolkien','Wladca');                // 2
  dodajK(ksiazki,1,' Prus','Lalka');                    // 3
  print(ksiazki);                                       // 4
  writeln;
  dodajS(studenci,'Kowalski');                           // 5
  dodajS(studenci,'Nowak');                             // 6
  dodajS(studenci,'Maj');                               // 7
  wyporycz(ksiazki,studenci,'Majaa','Bez mojej zgody'); // 8
  wyporycz(ksiazki,studenci,'Maj','Wladca');            // 9
  wyporycz(ksiazki,studenci,'Maj','Bez mojej zgody');   // 10
  wyporycz(ksiazki,studenci,'Maj','Bez mojej zgodcdcvy');// 11
  wyporycz(ksiazki,studenci,'Kowalski','Lalka');       // 12
  wyporycz(ksiazki,studenci,'Kowalski','Wladca');      // 13
  wyswietlS(studenci);                                  // 14
  readln;
end.
```

Wynik działania powyższego kodu:

```
0 Joudi Bez mojej zgody
1 Prus Lalka
2 Tolkien Wladca
```

Brak studenta

nie dodano

Brak ksiazki

nie dodano

Maj

```
0 Joudi Bez mojej zgody
```

```
2 Tolkien Wladca
```

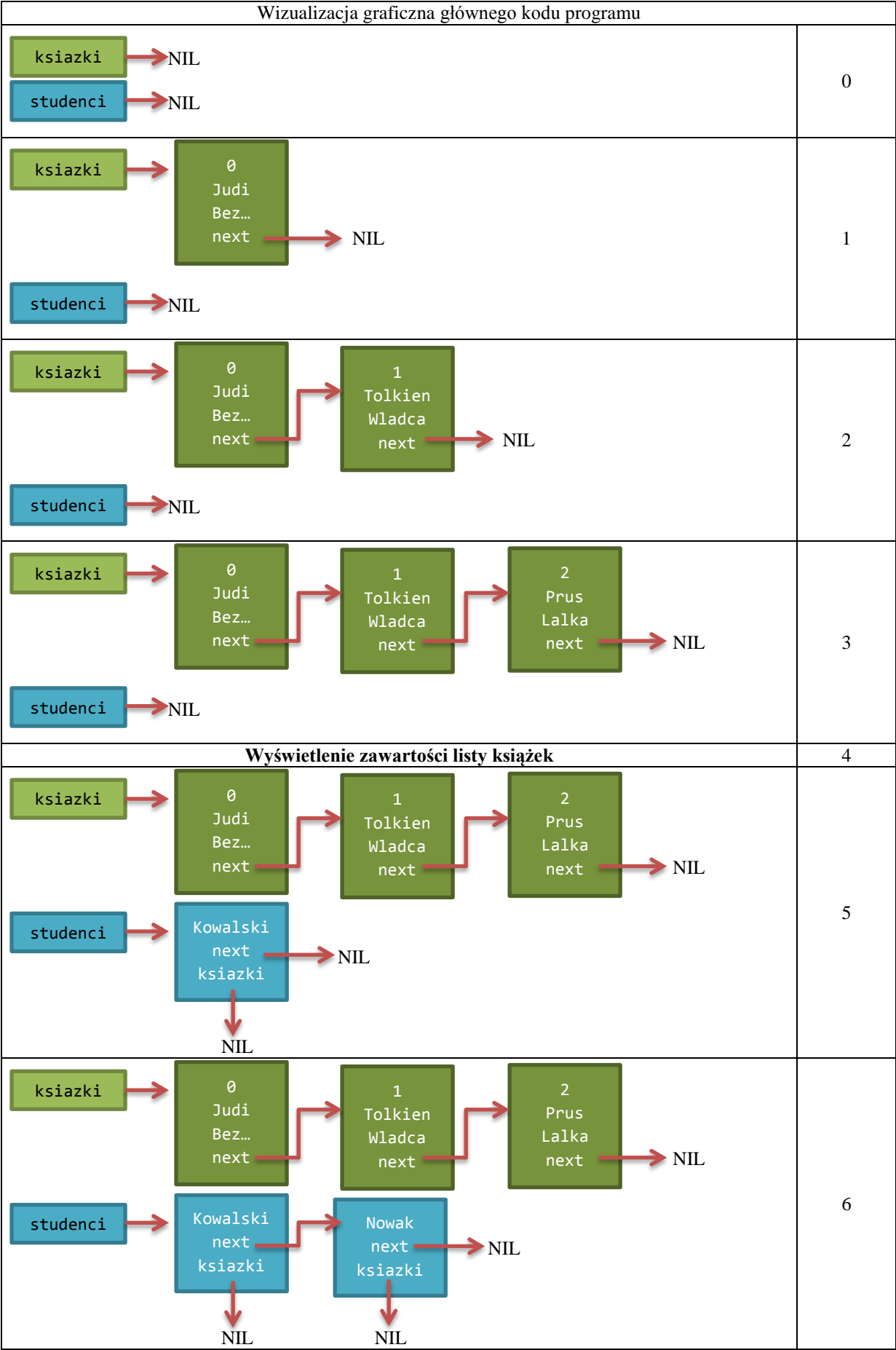
Nowak

Student nie czyta

Kowalski

```
1 Prus Lalka
```

```
2 Tolkien Wladca
```



	7
Wyświetlenie komunikatu – student nie istnieje + nie dodano	
	9
	10
Wyświetlenie komunikatu –brak książki + nie dodano	

	12
	13
Wyświetlenie listy studentów wraz z wypożyczonymi książkami	14

Zadania dodatkowe (do samodzielnej implementacji):

- Modyfikacja procedury wypożycz – po wypożyczeniu książka „znika” z listy książek
- Dodanie funkcji ZWRÓĆ – „zwracającej” książkę na listę dostępnych książek
- Dodanie zapisu/odczytu do pliku obu list
- Modyfikacja programu głównego – wczytanie obu list + menu (**case** – wybór jednej z dostępnych procedur lub zakończenie programu) + zapis na koniec działania programu;