Politechnika Wrocławska

Wstęp do programowania

INP001213Wcl

rok akademicki 2018/19

semestr zimowy

Wykład 8

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

A-1 p. 411B

Politechnika Wrocławska

Plan prezentacji

- Podział kodu programu
- Struktury
  - definiowanie struktur
  - dostęp do składowych struktur
  - wskaźniki
  - przykłady
- typedef
- Struktury cykliczne
  - lista
  - drzewo

Na podstawie:

- G. Perry, D. Miller, *Język C Programowanie dla początkujących*

Politechnika Wrocławska

Pliki nagłówkowe

- W złożonych programach istnieje konieczność rozdzielenia kodu źródłowego na kilka plików

Politechnika Wrocławska

Pliki nagłówkowe

main.c

```c
/*projekt01*/
/*Program ilustruje podział kodu źródłowego
programu między kilka plików.*/
#include <stdio.h>

//deklaracja funkcji - tą część można umieścić w pliku nagłówkowym
double max(double t[], int n);

#define N 10

//kod źródłowy funkcji głównej - pozostaje w głównym pliku programu
int main()
{
    double t[N] = {0.943, 0.464, 0.982, 0.70                0, 0.619, 0.517};
    printf("max: %.3f\n",max(t,N));
    return 0;
}

//kod źródłowy funkcji pomocniczej - można umieścić w osobnym pliku źródłowym
double max(double t[], int n){
    double m = t[0];
    int i;
    for(i=1; i<n; i++){
        if(t[i] > m)
            m = t[i];
    }
    return m;
}
```

deklaracje funkcji

funkcja główna

kod źródłowy funkcji

Politechnika Wrocławska

Pliki nagłówkowe

main.c

```c
/*projekt02*/
/*Program ilustruje podział kodu źródłowego
programu między kilka plików.*/
#include <stdio.h>
#include "sort.h" //dołączono plik nagłówkowy

#define N 10

//kod źródłowy funkcji głównej - pozostaje w głównym pliku programu
int main()
{
    double t[N] = {0.943, 0.464, 0.982, 0.709, 0.898, 0.167, 0.386, 0.770, 0.619, 0.517};
    printf("max: %.3f\n",max(t,N));
    return 0;
}
```

Politechnika Wrocławska

Pliki nagłówkowe

main.c | **sort.h**

```
1  /*projekt02*/
2  #ifndef SORT_H_INCLUDED
3  #define SORT_H_INCLUDED
4
5  //deklaracja funkcji
6  double max(double t[], int n);
7
8  #endif // SORT_H_INCLUDED
9
```

sort.h

Pliki nagłówkowe

main.c | sort.h | **sort.c**

```c
/*projekt02*/
//kod źródłowy funkcji pomocniczej
double max(double t[], int n){
    double m = t[0];
    int i;
    for(i=1; i<n; i++){
        if(t[i] > m)
            m = t[i];
    }
    return m;
}

```

sort.c

Politechnika Wrocławska

Struktury

- Tablice pozwalają przechowywać wiele wartości, ale wszystkie muszą być tego samego typu
- Jeżeli chcemy przechowywać dane różnych typów dotyczące jednego obiektu możemy wykorzystać strukturę

Politechnika Wrocławska

Struktury

- Przykład dane dotyczące studenta:
  - imię (tablica znaków),
  - nazwisko (tablica znaków),
  - numer indeksu (liczba całkowita),
  - rok studiów (liczba całkowita),
  - średnia ocen (liczba rzeczywista),
  - …

Struktury

Definicja

- W celu korzystania ze struktury należy zdefiniować jej składowe

```
struct [etykieta struktury]{
    definicja składowej;
    definicja składowej;
    ...
    definicja składowej;
};
```

Politechnika Wrocławska

Struktury

Definicja

- Przykładowa definicja struktury reprezentującej punkt na płaszczyźnie 2D

```
struct point{
    double x;
    double y;
};
```

Politechnika Wrocławska

Struktury

Definicja

- Definicja struktury określa układ składowych w strukturze
- Zdefiniowanie struktury nie jest równoznaczne z utworzeniem zmiennej
- Zmienne typu strukturalnego są deklarowane analogicznie jak zmienne typów podstawowych

```
struct point a, b;
```

Struktury

Dostęp do składowych

- Dostęp do składowych struktury uzyskuje się przy pomocy operatora . (kropka)

```
nazwaZmiennejStrukturalnej.nazwaSkładowej
```

- Zmienne typu strukturalnego są deklarowane analogicznie jak zmienne typów podstawowych

```
struct point a, b;
```

Struktury

Przykład

main.c | point.h

```
1   /*point.h*/
2   /*Plik nagłówkowy zawierający
3   definicję struktury point.*/
4   #ifndef POINT_H_INCLUDED
5   #define POINT_H_INCLUDED
6
7   //struktura point ma dwa pola
8   //typu double
9   struct point {
10      double x; //współrzędna x
11      double y; //współrzędna y
12  };
13
14  #endif // POINT_H_INCLUDED
```

Struktury

Przykład

main.c | point.h

```c
/*point.h*/
/*Plik nagłówkowy zawie
definicję struktury poi
#ifndef POINT_H_INCLUDE
#define POINT_H_INCLUDE

//struktura point ma dw
//typu double
struct point {
    double x; //współrzę
    double y; //współrzę
};

#endif // POINT_H_INCLU
```

main.c | point.h

```c
/*main.c*/
/*Program demonstruje wykorzystanie
struktury point zdefiniowanej
w pliku nagłówkowym point.h.*/

#include <stdio.h>
#include "point.h" //dołączenie pliku nagłówkowego
//nazwa pliku umieszczona w cudzysłowie

int main()
{
    //deklaracja zmiennych a, b
    //będących strukturami
    struct point a, b;

    //przypisanie wartości polom struktury a ...
    a.x =  1.1;
    a.y = -2.0;

    //... oraz b
    b.x = 0;
    b.y = 0;

    //wyświetlenie wartości pól struktury
    printf("a.x = %lf\n", a.x);
    printf("a.y = %lf\n", a.y);
    printf("b.x = %lf\n", b.x);
    printf("b.y = %lf\n", b.y);

    return 0;
}
```

Struktury

Przykład

main.c | point.h

```c
/*point.h*/
/*Plik nagłówkowy zawie
definicję struktury poi
#ifndef POINT_H_INCLUDE
#define POINT_H_INCLUDE

//struktura point ma dw
//typu double
struct point {
    double x; //współrzę
    double y; //współrzę
};

#endif // POINT_H_INCLU
```

```
a.x = 1.100000
a.y = -2.000000
b.x = 0.000000
b.y = 0.000000
```

main.c | point.h

```c
/*main.c*/
/*Program demonstruje wykorzystanie
struktury point zdefiniowanej
w pliku nagłówkowym point.h.*/

#include <stdio.h>
#include "point.h" //dołączenie pliku nagłówkowego
//nazwa pliku umieszczona w cudzysłowie

int main()
{
    //deklaracja zmiennych a, b
    //będących strukturami
    struct point a, b;

    //przypisanie wartości polom struktury a ...
    a.x =  1.1;
    a.y = -2.0;

    //... oraz b
    b.x = 0;
    b.y = 0;

    //wyświetlenie wartości pól struktury
    printf("a.x = %lf\n", a.x);
    printf("a.y = %lf\n", a.y);
    printf("b.x = %lf\n", b.x);
    printf("b.y = %lf\n", b.y);

    return 0;
}
```

Struktury

Wskaźniki

- Ze względu na to, że struktury mogą mieć duży rozmiar, korzystne może być posługiwanie się wskaźnikami na struktury
- Korzystne może być przekazywanie struktur do funkcji przez adres
- Dostęp do struktur wskazywanych przez wskaźnik uzyskuje się operatorem ->

```
wskaźnikNaStrukturę->nazwaSkładowej
(*wskaźnikNaStrukturę).nazwaSkładowej
```

Struktury

Przykład

main.c | point.h | point.c

```
1  /*point.h*/
2  /*Plik nagłówkowy zawierający
3  definicję struktury point
4  oraz deklaracje obsługujących ją
5  funkcji.*/
6  #ifndef POINT_H_INCLUDED
7  #define POINT_H_INCLUDED
8
9  struct point {
10    double x;
11    double y;
12 };
13
14 //w pliku nagłówkowym dodano deklaracje
15 //dwóch funkcji obsługujących strukturę point:
16     //funkcja, która wczytuje współrzędne od użytkownika
17 void readPoint(struct point*);
18     //funkcja, która wyświetla współrzędne na ekran
19 void printPoint(struct point*);
20 //obie funkcje pobierają argument przez wskaźnik
21
22 #endif // POINT_H_INCLUDED
```

Struktury

Przykład

main.c | point.h | **point.c**

```c
/*point.c*/
/*Plik źródłowy zawierający
definicje funkcji obsługujących
strukturę point.*/

//wykorzystanie funkcji printf(), scanf()
//wymaga dołączenia pliku stdio.h
#include<stdio.h>
//dołączamy plik nagłówkowy,
//kompilując funkcje pracujące na strukturze
//musimy znać jej definicję
#include"point.h"

void readPoint(struct point* p){

    printf("x coordinate: ");
    scanf("%lf",&(p->x));
    //&(p->x) - zapis oznacza adres składowej x
    //w strukturze wskazywanej przez wskaźnik p
    // p      - wskaźnik na strukturę typu point
    // p->x   - pole x w strukutrze wskazywanej przez p
    // (operator strzałka pozwala na dostęp do pól
    // wskazywanej struktury
    //&(p->x) - adres pola x

    printf("y coordinate: ");
    scanf("%lf",&(p->y));
}

void printPoint(struct point* p){
    printf("[%f,%f]",p->x,p->y);
}
```

Struktury

Przykład

main.c | point.h | point.c

```c
/*main.c*/
/*Program demonstruje wykorzystanie
struktury point zdefiniowanej
w pliku nagłówkowym point.h
wraz z funkcjami zdefiniowanymi
w pliku point.c.*/

#include <stdio.h>
#include "point.h"

int main()
{
    struct point a, b;

    //wczytanie współrzędnych punktów
    //z wykorzystaniem funkcji readPoint()
    readPoint(&a);
    readPoint(&b);

    //wyświetlenie współrzędnych punktów
    //z wykorzystaniem funkcji printPoint()
    printf("a = ");
    printPoint(&a);
    printf("\n");
    printf("b = ");
    printPoint(&b);
    printf("\n");

    return 0;
}
```

```
x coordinate: 1
y coordinate: 2
x coordinate: 5
y coordinate: 3
a = [1.000000,2.000000]
b = [5.000000,3.000000]
```

Politechnika Wrocławska

Struktury

- Strukturę można inicjalizować, umieszczając po definicji listę wartości początkowych

```
struct point a = {3.20, 2.00};
```

Politechnika Wrocławska

Struktury

- Można definiować struktury zawierające inne struktury
- Przykładowo informacje o trójkącie można przechowywać w strukturze zdefiniowanej następująco

```
struct triangle {
    struct point v0;
    struct point v1;
    struct point v2;
};
```

Politechnika Wrocławska

Struktury

- Trzy pola można zastąpić tablicą struktur

```
struct triangle {
    struct point vertices[3];
};
```

Politechnika Wrocławska

Struktury

- Podobnie można zdefiniować strukturę opisującą wielokąt

```
struct polygon {
    int n;
    struct point * vertices;
};
```

Politechnika Wrocławska

typedef

- Język C dysponuje mechanizmem, który umożliwia tworzenie nowych nazw typów danych

```
typedef okreslenie_typu nowa_nazwa;
```

```
typedef int Length;
```

```
Length len, maxlen;
Length *lengths[];
```

Politechnika Wrocławska

typedef

main.c | point.h | point.c

```
1   /*point.h*/
2   /*Plik nagłówkowy zawierający
3   definicję struktury point
4   oraz deklaracje obsługujących ją
5   funkcji.*/
6   #ifndef POINT_H_INCLUDED
7   #define POINT_H_INCLUDED
8
9   struct point {
10      double x;
11      double y;
12  };
13
14  typedef struct point point;
15
16  //w pliku nagłówkowym dodano deklaracje
17  //dwóch funkcji obsługujących strukturę point:
18      //funkcja, która wczytuje współrzędne od użytkownika
19  void readPoint(struct point*);
20      //funkcja, która wyświetla współrzędne na ekran
21  void printPoint(struct point*);
22  //obie funkcje pobierają argument przez wskaźnik
23
24  #endif // POINT_H_INCLUDED
```

Politechnika Wrocławska

Struktury cykliczne

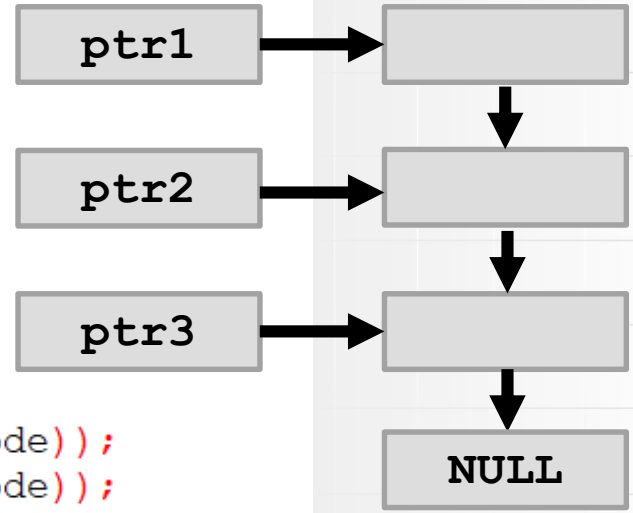
- Polem struktury może być wskaźnik na taką strukturę – tego typu struktury nazywamy strukturami cyklicznymi
- Polem struktury nie może być sama struktura

Struktury cykliczne

Przykład

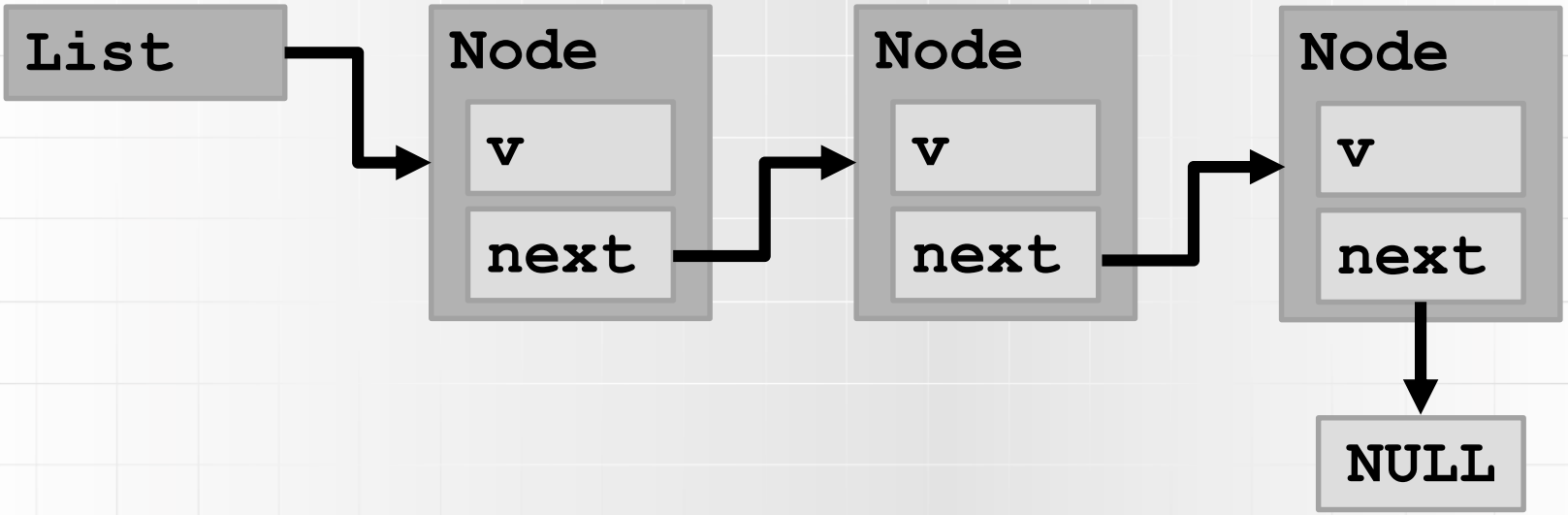
Start here | node.c

```c
/*node.c*/
/*Przykładowy program zawierający definicję
struktury cyklicznej*/
#include <stdlib.h>

struct Node{
    struct Node * ptr;
};

typedef struct Node Node;

int main(){
    Node * ptr1 = malloc(sizeof(Node));
    Node * ptr2 = malloc(sizeof(Node));
    Node * ptr3 = malloc(sizeof(Node));
    ptr1->ptr = ptr2;
    ptr2->ptr = ptr3;
    ptr3->ptr = NULL;
    free(ptr1);
    free(ptr2);
    free(ptr3);
    return 0;
}
```



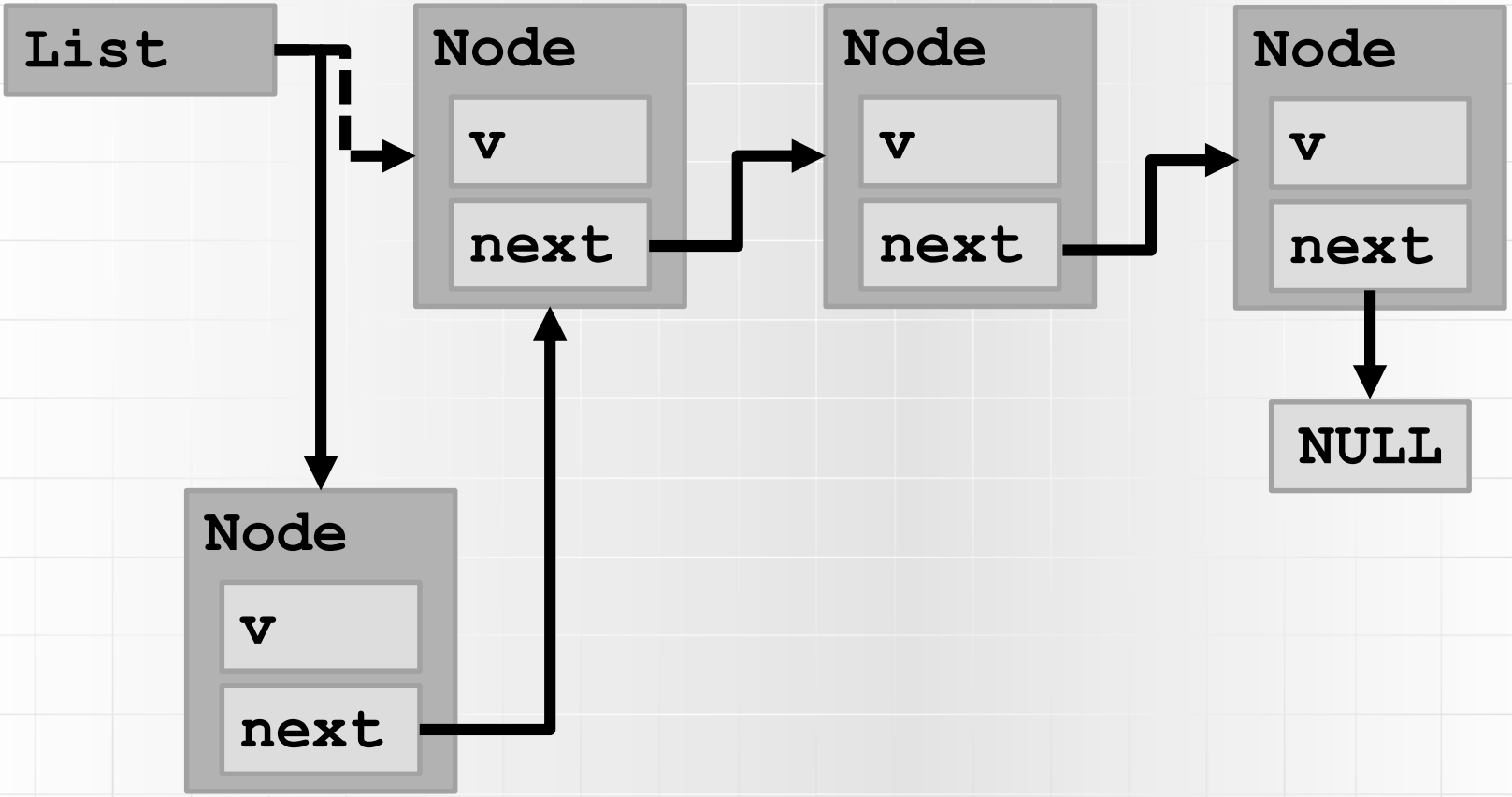
Politechnika Wrocławska

Lista



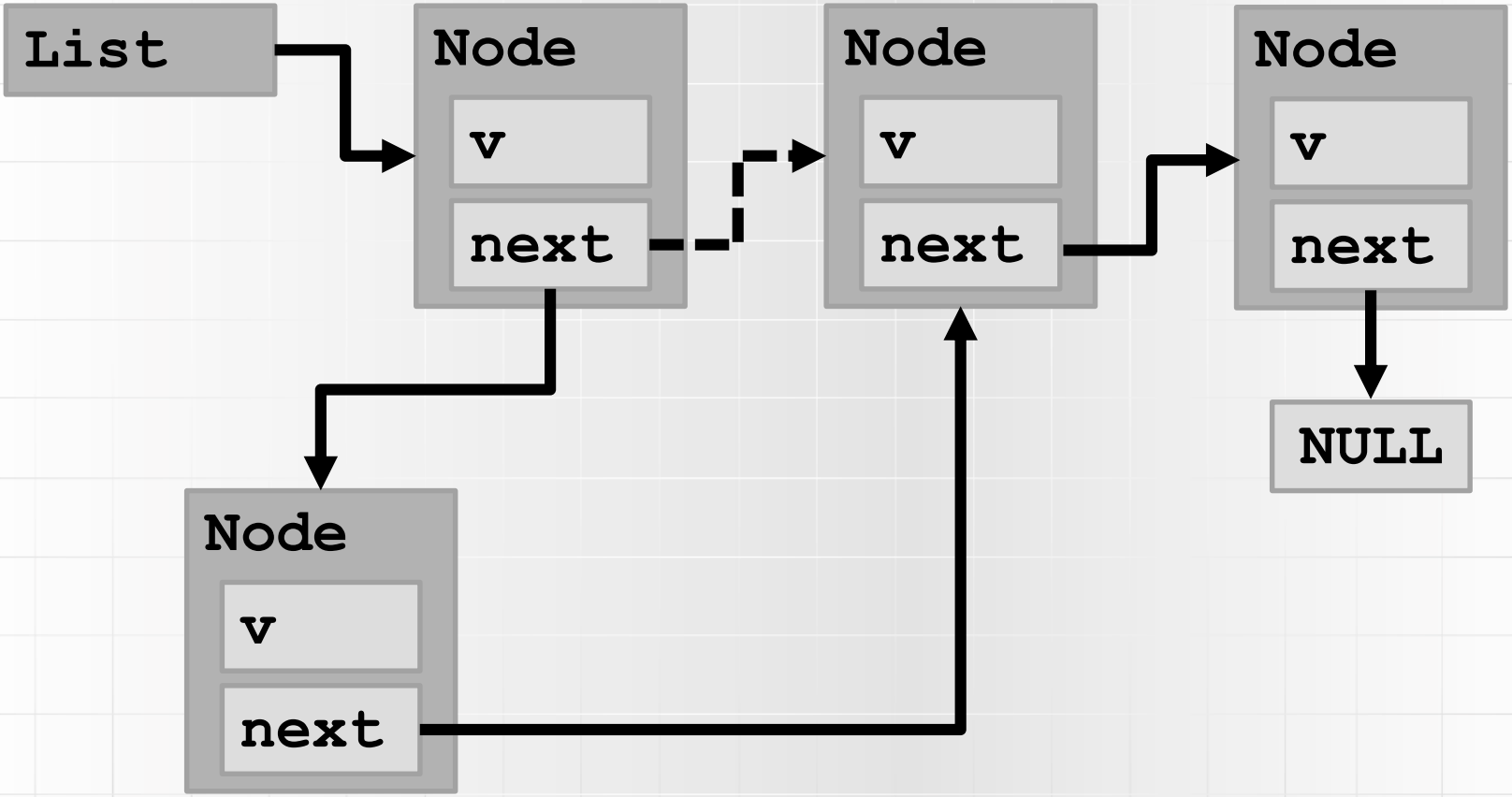
Lista

Dodaj element (na początku)



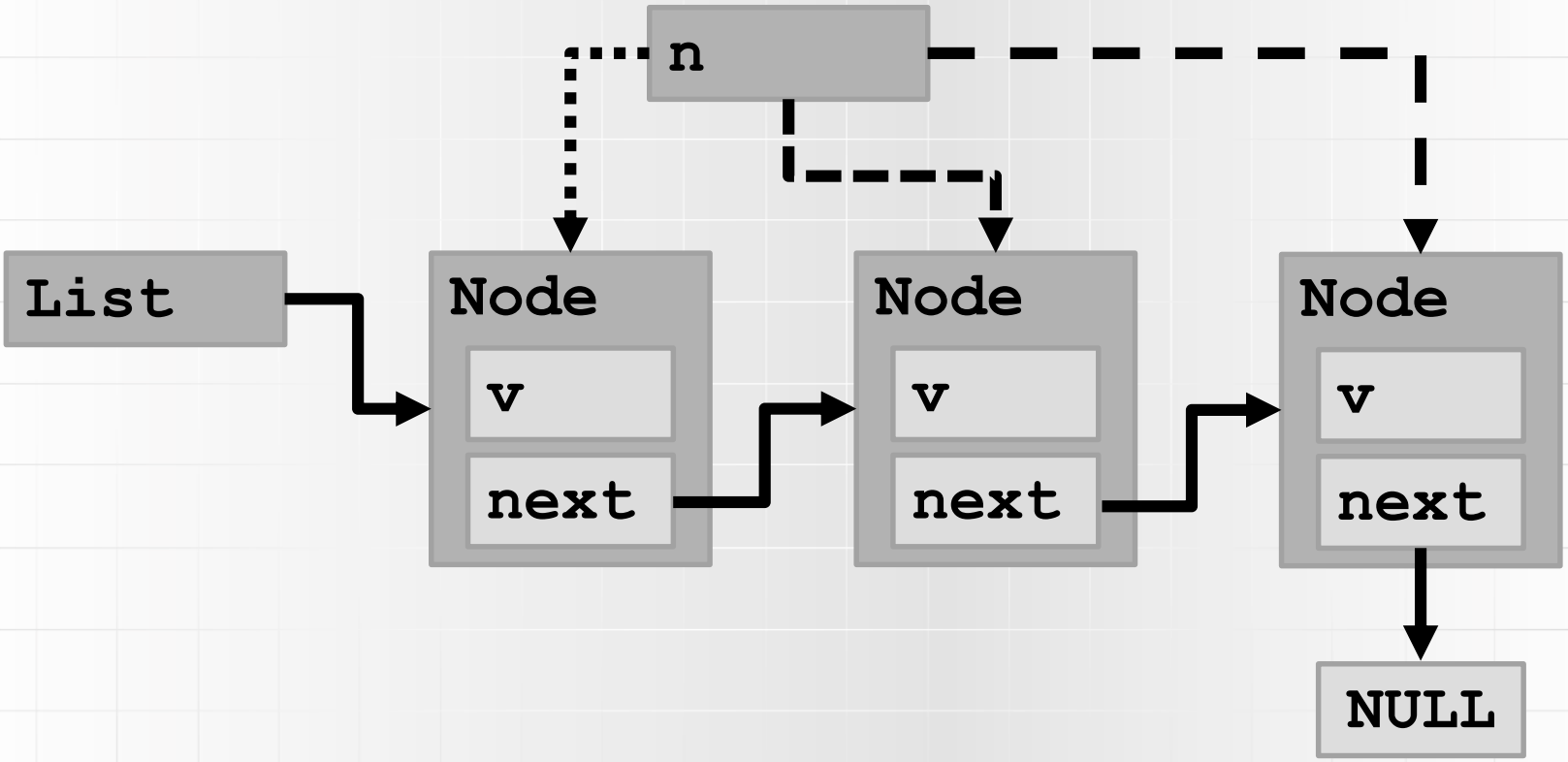
Lista

Dodaj element (wewnątrz)



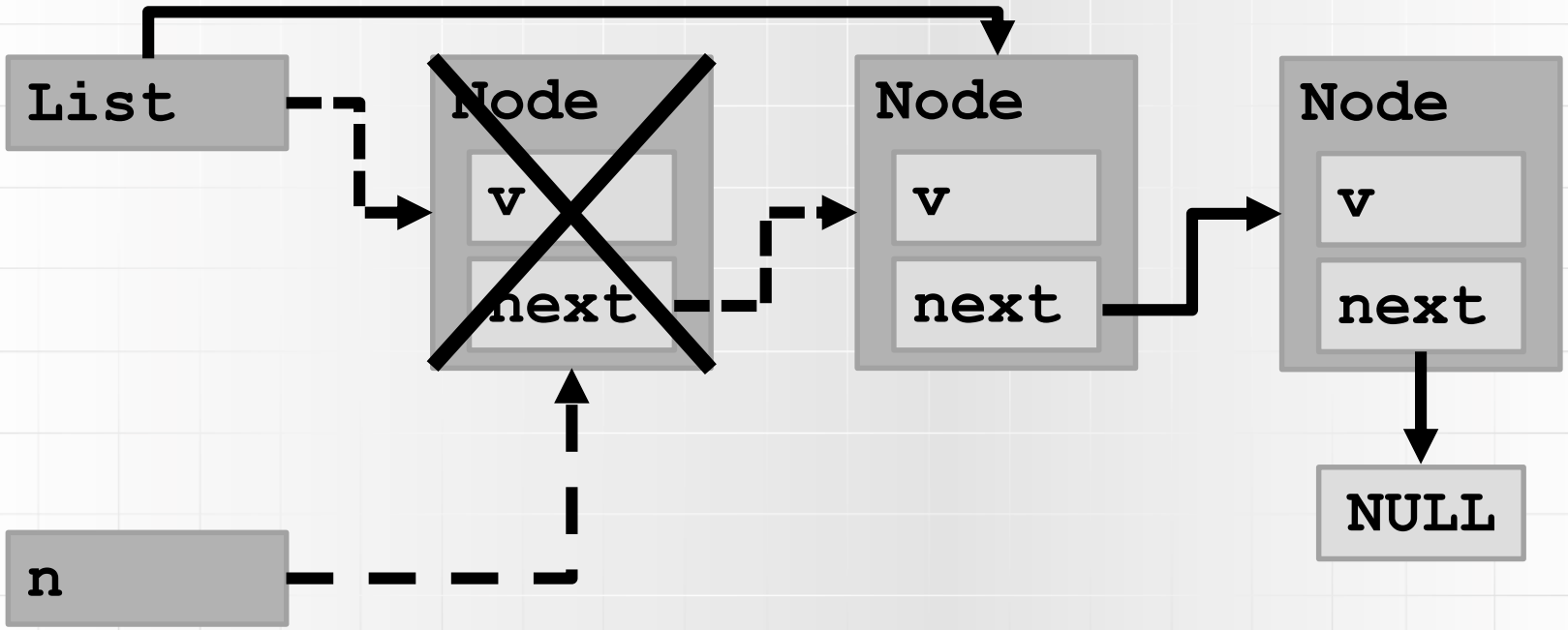
Lista

Wypisz (iteracyjnie)



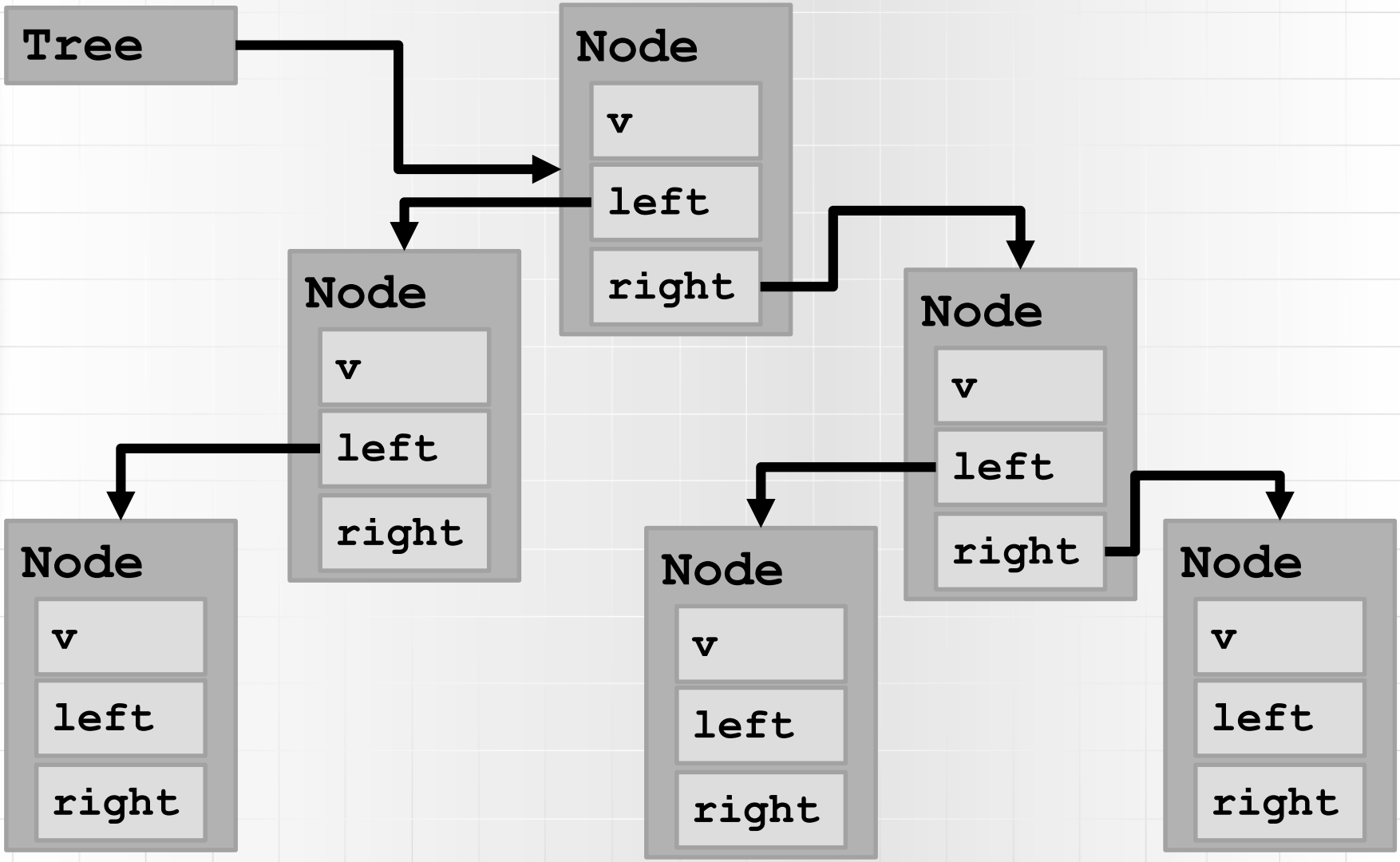
Lista

Usuń pierwszy element



Politechnika Wrocławska

Drzewo



Politechnika Wrocławska

Podsumowanie

- Rozdzielanie programu na kilka plików źródłowych
- Struktury danych
  - definiuj struktury, gdy chcesz zgrupować dane
  - zdefiniuj strukturę, zanim zadeklarujesz zmienną jej typu
  - dostęp do składowych uzyskuje się przy użyciu operatora . (kropka)
  - operator -> służy do odwoływania się do składowej struktury wskazywanej zmienną wskaźnikową
  - przykładowe złożone struktury danych: lista, drzewo