

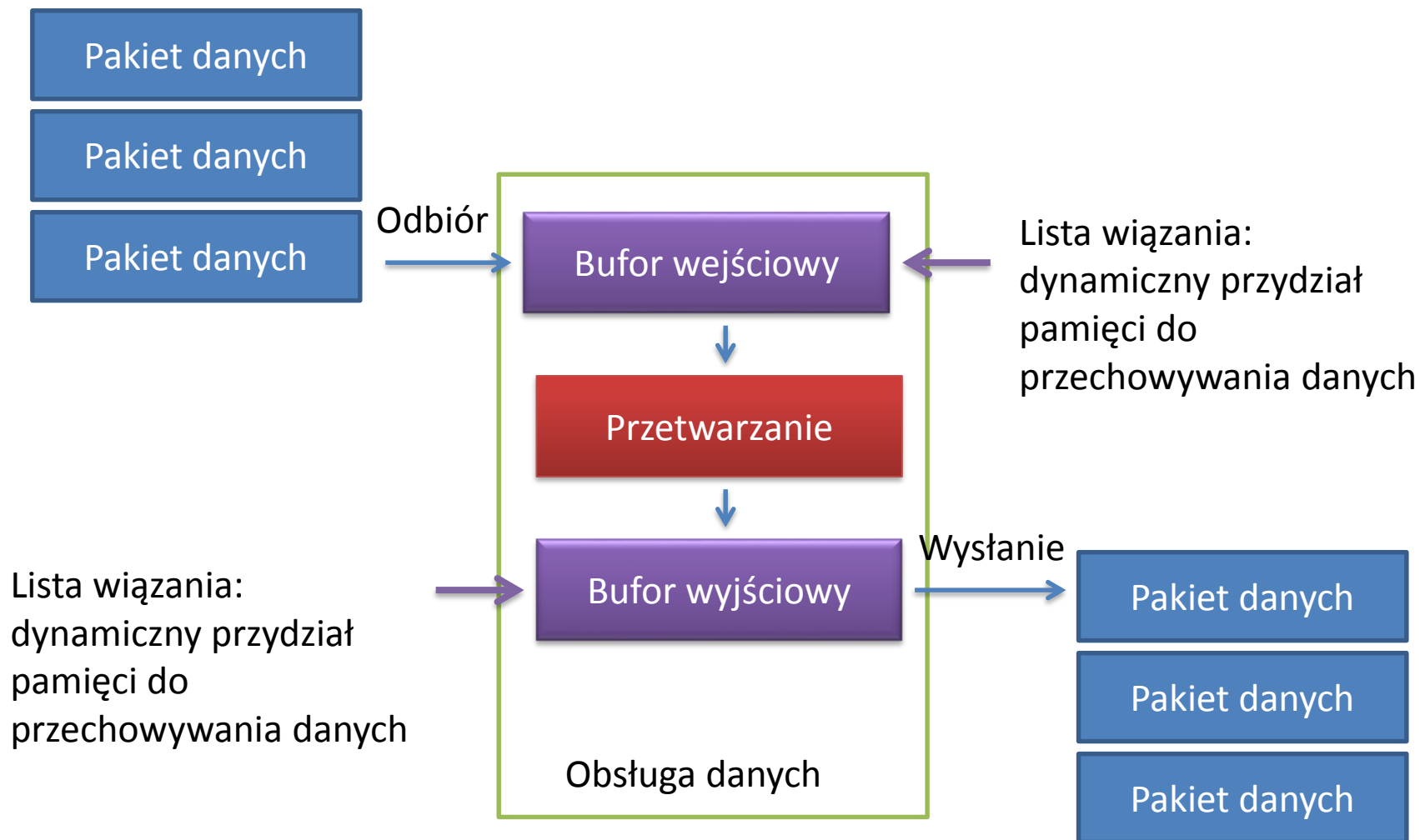
PRZEDMIOT
„PROGRAMOWANIE W JĘZYKU C/C++ DLA ZASTOSOWAŃ
SIECIOWYCH – CZ.2”
STACJONARNE STUDIA I°

T1: Zaawansowane programowanie w języku C/C++

dr inż. Jarosław KRYGIER
p. 122 b.47, tel. 22 6837193
email: jkrygier@wat.edu.pl

Materiały dydaktyczne: jkrygier.wel.wat.edu.pl

Lista wiązana – po co?

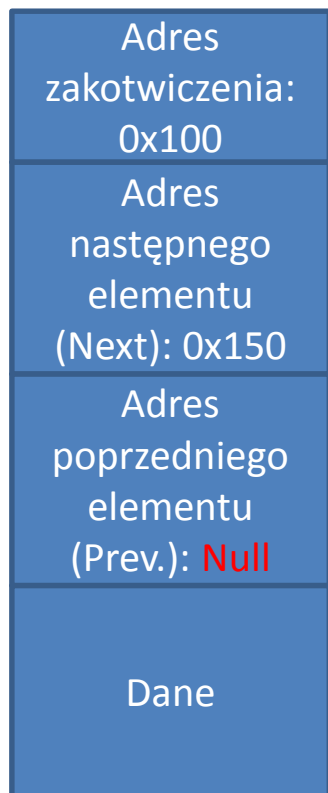


Lista wiązana dwukierunkowa

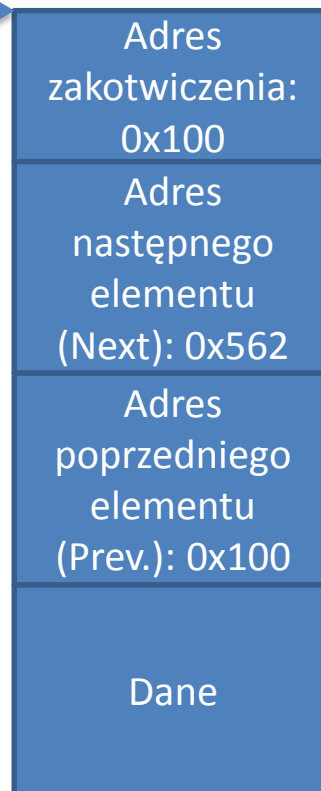
Adres zakotwiczenia listy (pierwszego elementu): 0x100

Adres:
0x100

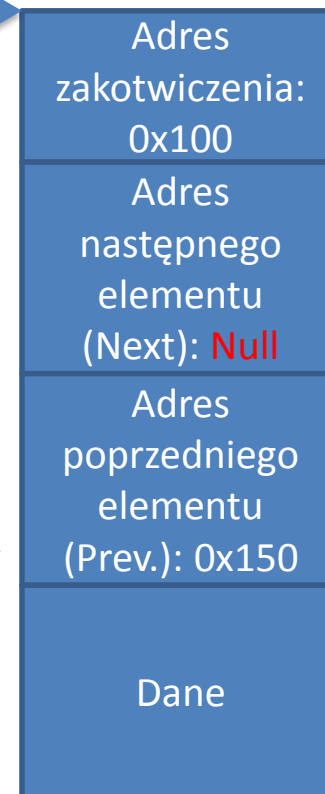
Element
listy



Adres:
0x150



Adres:
0x562



Element listy (przykład1)

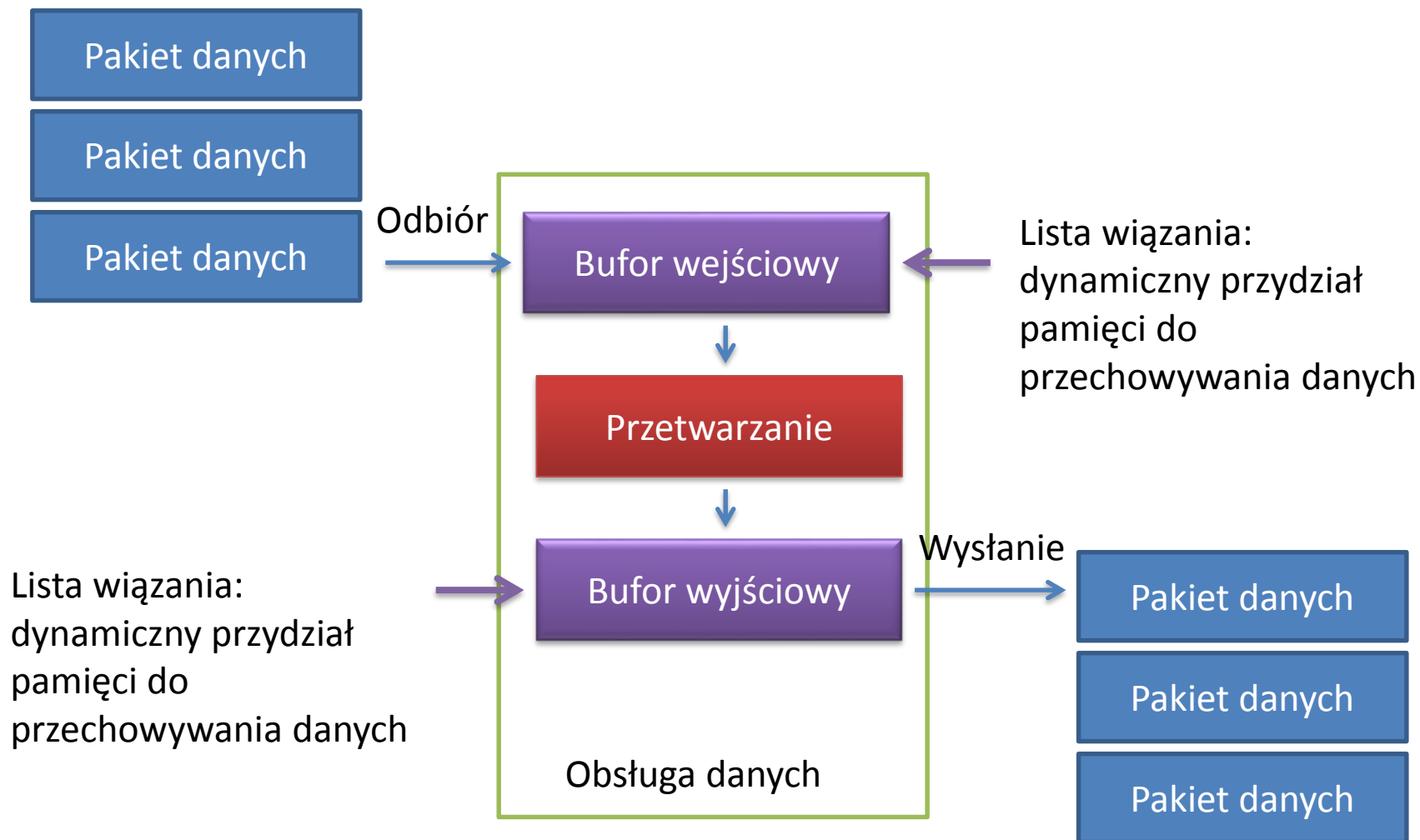
```
struct element {  
    struct element *nastepny; /// wskaźnik na następny element listy  
    struct element *poprzedny; /// wskaźnik na poprzedni element listy  
    struct element *pierwszy; /// wskaźnik na pierwszy element listy  
    unsigned char pakiet[1500]; /// bufor znakowy  
};
```

Element listy (przykład2)

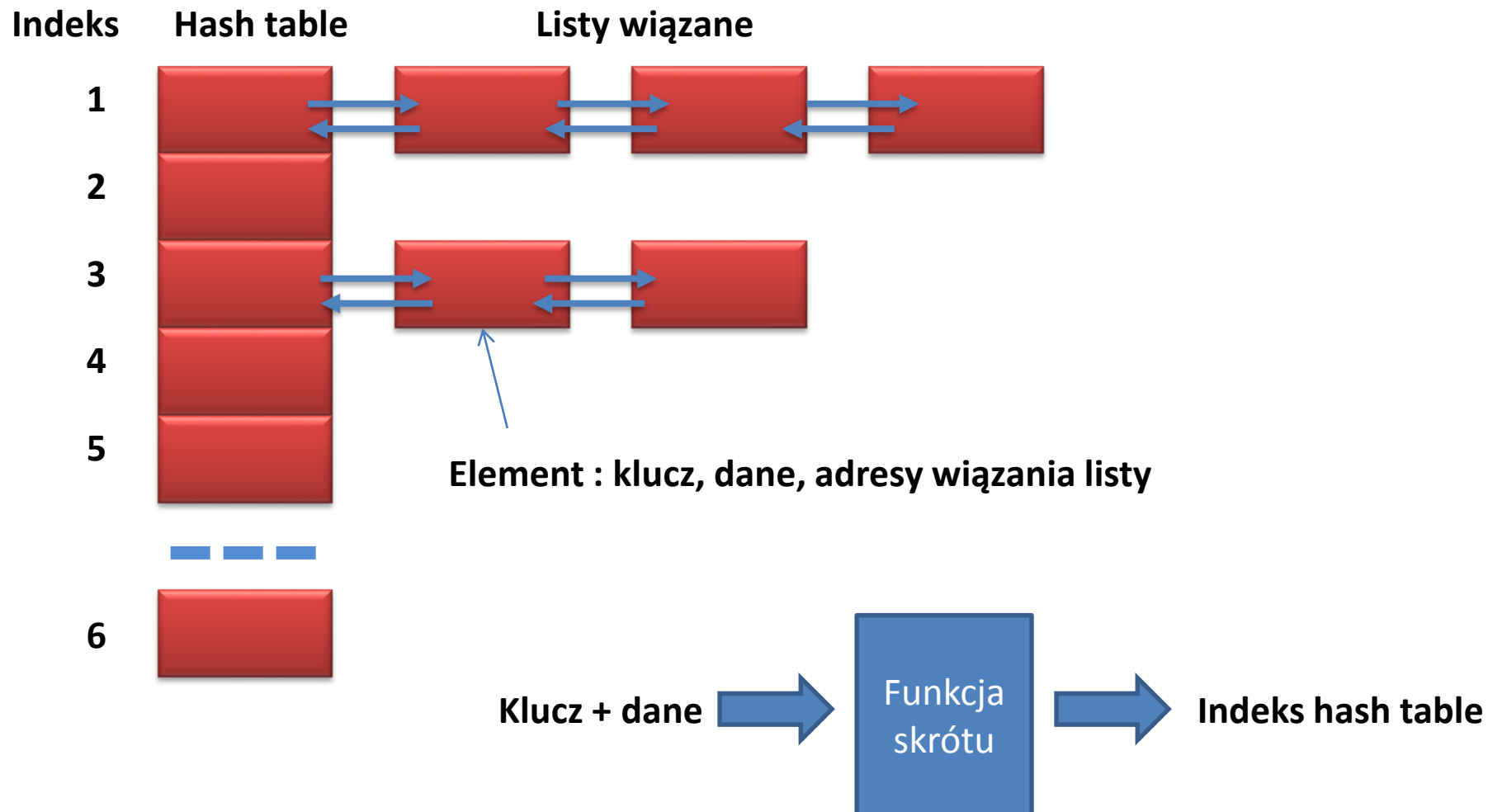
```
struct element {  
    struct element *nastepny; ///wskaźnik na następny element listy  
    struct element *poprzedny; ///wskaźnik na poprzedni element listy  
    struct element *pierwszy; ///wskaźnik na pierwszy element listy  
    struct pakiet  pakiet; ///struktura z pakietem danych  
};
```

```
struct pakiet{  
    struct naglowek_pk;  
    unsigned char dane_pk[500];  
};
```

Lista wiązana – po co?



Lista wiązana – po co?

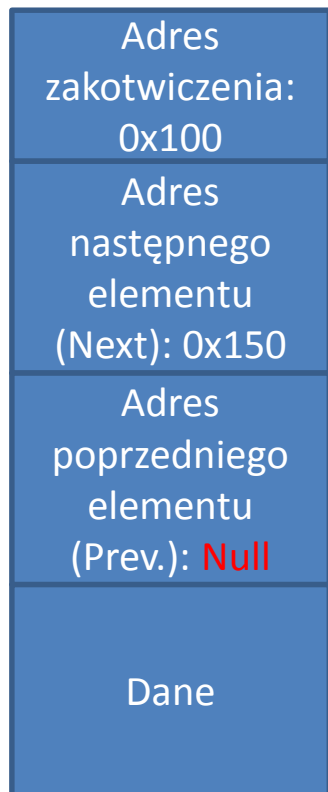


Lista wiązana dwukierunkowa

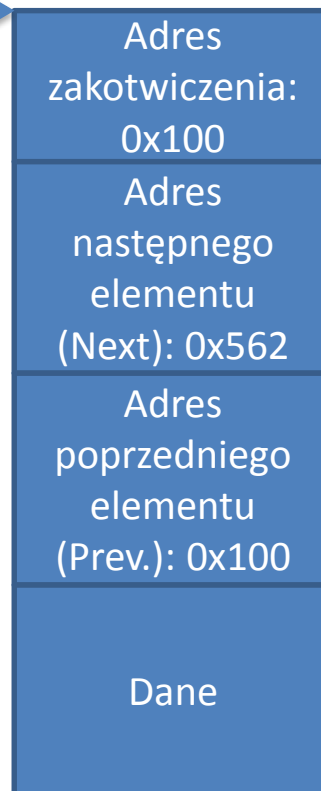
Adres zakotwiczenia listy (pierwszego elementu): 0x100

Adres:
0x100

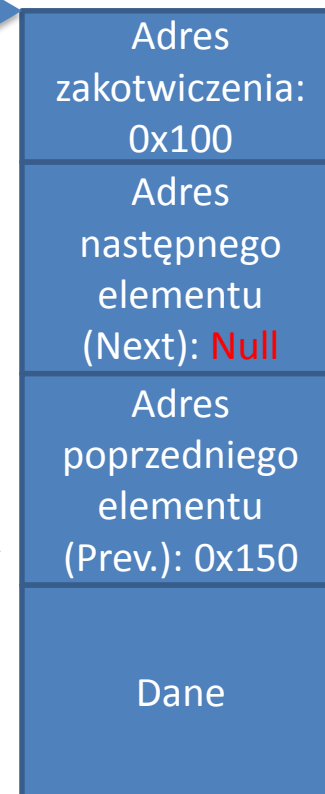
Element
listy



Adres:
0x150



Adres:
0x562



Element listy (przykład1)

```
struct element {  
    struct element *nastepny; ///wskaźnik na następny element listy  
    struct element *poprzedny; ///wskaźnik na poprzedni element listy  
    struct element *pierwszy; ///wskaźnik na pierwszy element listy  
    unsigned char pakiet[1500]; ///bufor znakowy  
};
```

Deklaracja wskaźnika na funkcję

```
int (*funkcja) (int, int);
```

Można:

- przekazywać do innych funkcji jako argumenty
- zwracać jako wynik działania funkcji
- porównywać z NULL

Nie można

- wykonywać operacji matematycznych

Biblioteki dynamiczne

//Utworzenie biblioteki dynamicznej:

//gcc -shared biblioteka.c -o biblioteka.so

Aby mieć 'dynamic library' (dl), linker musi łączyć z opcja -l: dl

Ustawić w Eclipse: C/C++ -> Settings -> GCC C Linker -> Libraries

Program może ładować w trakcie pracy potrzebne mu biblioteki. Linker dostarcza w tym celu 4 funkcji:

dlopen ładowanie biblioteki,

dlsym zwrócenie wskaźnika do odpowiedniego symbolu (funkcji) w bibliotece,

dlerror obsługa błędów,

dlclose zamykanie biblioteki.

Funkcje te są udostępniane poprzez nagłówek dlfcn.h (trzeba go dodać do głównego programu).

#include <dlfcn.h>

void *dlopen(const char *filename, int flag);

char *dlerror(void);

void *dlsym(void *handle, const char *symbol);

int dlclos(void *handle);