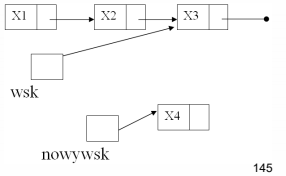
ISO/ANSI C – zm. dynamiczne

Dodanie elementu na końcu listy:

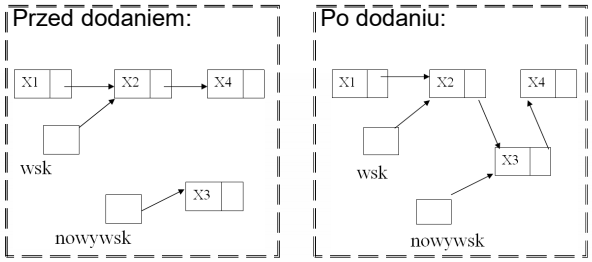
```
nowywsk= malloc(sizeof(struct film_t));
strcpy(nowywsk->tytul,"Kingsajz");
nowywsk->rok=1987;
nowywsk->nast = NULL;

wsk->nast = nowywsk;
wsk = wsk->nast;
```



145

ISO/ANSI C – zm. dynamiczne

Dodanie do listy uporządkowanej (np.wg roku)



146

ISO/ANSI C – zm. dynamiczne

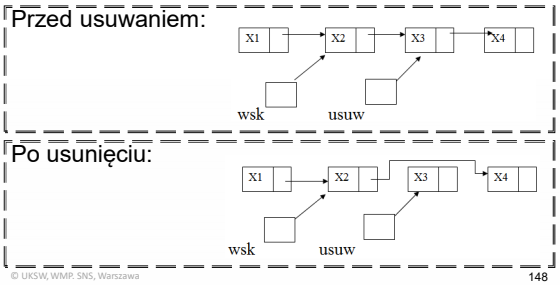
Dodanie do listy uporządkowanej malejąco (np. wg roku)

```
nowywsk = malloc(sizeof(struct film_t));
strcpy(nowywsk->tytul,"Bogowie");
nowywsk->rok=2014;
nowywsk->nast = NULL;

if(glowa==NULL)
    glowa = wsk = nowywsk;
else
if (glowa->rok < nowywsk->rok) {
    nowywsk->nast = glowa;
    glowa = nowywsk;
}
else
    if (glowa->nast == NULL)
        glowa->nast = nowywsk;
else {
    wsk = glowa;
    while (wsk->nast->rok > nowywsk->rok) {
        wsk = wsk->nast;
        if (wsk->nast == NULL)
            break;
    };
    nowywsk->nast = wsk->nast;
    wsk->nast = nowywsk;
}
```


147

ISO/ANSI C – zm. dynamiczne

Usuwanie wskazanego elementu z listy



148

ISO/ANSI C – zm. dynamiczne

Usuwanie wskazanego elementu z listy *(filmu z roku 2010)*

```
rok = 2010; // wskazuje na rok
            // filmu do usunięcia

if (glowa->rok == rok) {
    usuw = glowa;
    glowa = usuw->nast;
}
else {
    wsk = glowa;
    while ((wsk->nast != NULL)) {
        if (wsk->nast->rok == rok)
            break;
        wsk = wsk->nast;
    };
    if(wsk->nast != NULL) {   // znaleziono!
        usuw = wsk->nast;
        wsk->nast = wsk->nast->nast;
        free(usuw);
    };
}
```


149

ISO/ANSI C – zm. dynamiczne

Zaleta list jednokierunkowych:

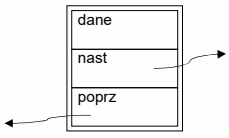
- zużywają pamięć proporcjonalnie do potrzeb

Wada list jednokierunkowych:

- wskaźniki mogą przesuwać się jedynie do przodu

Listy dwukierunkowe:

```
struct film_t {
    char tytul[80];
    int rok;
    struct film_t *nast;
    struct film_t *poprz;
};
```

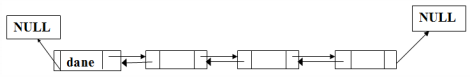


150

ISO/ANSI C – zm. dynamiczne

Listy dwukierunkowe:

- możliwość poruszania się wskaźnika w obydwu kierunkach
- zużywają minimalnie więcej pamięci, są za to szybsze w dostępie do danych

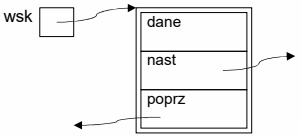


151

ISO/ANSI C – zm. dynamiczne

Listy dwukierunkowe:

```
struct film_t *wsk;
..
wsk = wsk->nast;  /* Przemieszczanie się w przód: */
..
wsk = wsk->poprz; /* Przemieszczanie się w tył: */
```



152

ISO/ANSI C – zm. dynamiczne

Listy dwukierunkowe – dodawanie:

`struct film_t *wsk, *nowy;`

nowy – wskazuje na nowy element do umieszczenia

wsk – wskazanie na element po którym mamy umieścić 'nowy'

```
nowy->nast = wsk->nast;
wsk->nast->poprz = nowy;
wsk->nast = nowy;
nowy->poprz = wsk;
```


153

ISO/ANSI C – zm. dynamiczne

Listy dwukierunkowe – dodawanie:

`struct film_t *wsk, *nowy;`

nowy – wskazuje na nowy element do umieszczenia

wsk – wskazanie na element przed którym mamy umieścić 'nowy'

```
nowy->nast = wsk;
nowy->poprz = wsk->poprz;
wsk->poprz->nast = nowy;
wsk->poprz = nowy;
```


154