Pracownia programowania - Wykład 14

Listy jednokieronowe - wybrane przypadki

Repozytoria

- Krzaczkowski
  - https://github.com/pjastr/Listy
  - https://github.com/pjastr/ListyJednokierunkowe2023
- Drugi zbiór:
  - https://github.com/pjastr/zbior-zadan-c-rozw

Lista bez głowy - wyświetlanie

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void printListWithoutHead(struct element *list) {
    if (list == NULL) {
        printf("Lista jest pusta\n");
        printf("---\n");
        return;
    }
    struct element *current = list;
    while (current != NULL) {
        printf("%d\n", current->x);
        current = current->next;
```

Lista z głową - wyświetlanie

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void printListWithHead(struct element *list) {
    if (list->next == NULL) {
        printf("Lista jest pusta\n");
        printf("---\n");
        return;
    }
    struct element *current = list->next;
    while (current != NULL) {
        printf("%d\n", current->x);
        current = current->next;
```

Lista bez głowy - tworzenie

```c
#include <stdio.h>

struct element {
    int x;
    struct element * next;
};

struct element * createNoHead() {
    return NULL;
}

int main() {
    struct element * list = createNoHead();
    printf("%p\n", list);
    return 0;
}
```

Lista z głową - tworzenie

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

struct element * createWithHead() {
    struct element * head = malloc(sizeof(struct element));
    head->next = NULL;
    return head;
}

int main() {
    struct element * list = createWithHead();
    printf("%p\n", list);
    printf("%p\n", list->next);
```

Lista bez głowy - dodanie na początek - pojedynczy wskaźnik

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

//rozwiazanie za pomoca pojedynczych wskaznikow
struct element * addFirst(struct element *list, int a) {
    struct element *newElement = malloc(sizeof(struct element));
    newElement->x = a;
    newElement->next = list;
    return newElement;
}

void printList(struct element *list) {
    while (list != NULL) {
```

Lista bez głowy - dodanie na początek - podwójny wskaźnik

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

//rozwiazanie za pomoca podwojnego wskaznika
void addFirst(struct element ** list, int a) {
    struct element *newElement = malloc(sizeof(struct element));
    newElement->x = a;
    newElement->next = *list;
    *list = newElement;
}

void printList(struct element *list) {
    while (list != NULL) {
```

Lista z głową - dodanie na początek

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void addFirst(struct element *list, int a) {
    struct element *newElement = malloc(sizeof(struct element));
    newElement->x = a;
    newElement->next = list->next;
    list->next = newElement;
}

void printList(struct element *list) {
    struct element *tmp = list->next;
    while (tmp != NULL) {
```

Lista bez głowy - dodanie na koniec - pojedynczy wskaznik

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

struct element * addLast(struct element *list, int a) {
    struct element *newElement = malloc(sizeof(struct element));
    newElement->x = a;
    newElement->next = NULL;
    if (list == NULL) {
        return newElement;
    }
    struct element *current = list;
    while (current->next != NULL) {
        current = current->next;
```

Lista bez głowy - dodanie na koniec - podwójny wskaznik

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void addLast(struct element ** list, int a) {
    struct element *newElement = malloc(sizeof(struct element));
    newElement->x = a;
    newElement->next = NULL;
    if (*list == NULL) {
        *list = newElement;
        return;
    }
    struct element *current = *list;
    while (current->next != NULL) {
```

Lista z głową - dodanie na koniec

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void addLast(struct element *list, int a) {
    struct element *current = list;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = malloc(sizeof(struct element));
    current->next->x = a;
    current->next->next = NULL;
}

```

Lista bez głowy - usunięcie całej listy

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

struct element * removeList(struct element * list) {
    struct element * current = list;
    struct element * next;
    while (current != NULL) {
        next = current->next;
        free(current);
        current = next;
    }
    return NULL;
}
```

Lista z głową - usunięcie całej listy

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void removeList(struct element * list){
    struct element * current = list->next;
    struct element * next;
    while(current != NULL){
        next = current->next;
        free(current);
        current = next;
    }
    list->next = NULL;
}
```

Lista bez głowy - usunięcie początkowego elementu

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

struct element * removeFirst(struct element * list) {
    if (list == NULL) {
        return NULL;
    }
    struct element * newHead = list->next;
    free(list);
    return newHead;
}

void printList(struct element * list) {
```

Lista z głową - usunięcie początkowego elementu

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void removeFirst(struct element *list) {
    if (list->next == NULL) {
        return;
    }
    struct element *temp = list->next;
    list->next = list->next->next;
    free(temp);
}

void printList(struct element *list) {
```

Lista bez głowy - usunięcie ostatniego elementu

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

struct element * deleteLast(struct element *list) {
    if (list == NULL) {
        return NULL;
    }
    if (list->next == NULL) {
        free(list);
        return NULL;
    }
    struct element *current = list;
    while (current->next->next != NULL) {
```

Lista z głową - usuniecie ostatniego elementu

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void removeLast(struct element * list) {
    if (list->next == NULL) {
        return;
    }
    struct element * prev = list;
    struct element * current = list->next;
    while (current->next != NULL) {
        prev = current;
        current = current->next;
    }
```

Lista bez głowy - usunięcie przedostatniego elementu

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

struct element * deleteBeforeLast(struct element *list) {
    if (list == NULL || list->next == NULL) {
        return list;
    }
    struct element *prev = NULL;
    struct element *current = list;
    struct element *next = list->next;
    while (next->next != NULL) {
        prev = current;
        current = next;
```

Lista z głową - usunięcie przedostatniego elementu

```c
#include <stdio.h>
#include <stdlib.h>

struct element {
    int x;
    struct element * next;
};

void removeBeforeLast(struct element * list) {
    if (list->next == NULL || list->next->next == NULL) {
        return;
    }
    struct element * prev = list;
    struct element * current = list->next;
    struct element * next = current->next;
    while (next->next != NULL) {
        prev = current;
        current = next;
```