**LISTY JEDNOKIERUNKOWE**

struct element

{

int i;

struct element \* next;

};  
**Tworzenie struktury listy.  
Pole i to wartość konkretngo elementu. Pole next to wskaźnik na następnik.**

**LISTA BEZ GŁOWY**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \*next;

};

int main()

{

**// stwórz listę bez głowy o elementach 5,7,-4**

struct element \* lista = malloc(sizeof(struct element));

lista->i = 5;

lista->next = malloc(sizeof(struct element));

lista->next->i = 7;

lista->next->next = malloc(sizeof(struct element));

lista->next->next->i = -4;

lista->next->next->next = NULL;

**//wyswietl listę**

struct element \* wsk = lista;

while(wsk != NULL)

{

printf("%d\n", wsk->i);

wsk = wsk->next;

}

return 0;

}

**LISTA Z GŁOWĄ**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \*next;

};

int main()

{

**// stwórz listę z głową o elementach 5,7,-4**

struct element \* lista = malloc(sizeof(struct element));

lista->next = malloc(sizeof(struct element));

lista->next->i = 5;

lista->next->next = malloc(sizeof(struct element));

lista->next->next->i = 7;

lista->next->next->next = malloc(sizeof(struct element));

lista->next->next->next->i = -4;

lista->next->next->next->next = NULL;

**//wyswietl listę**

struct element \* wsk = lista->next;

while(wsk != NULL)

{

printf("%d\n", wsk->i);

wsk = wsk->next;

}

return 0;

}

**PUSTA LISTA BEZ GŁOWY**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \*next;

};

int main()

{

struct element \* lista = NULL;

return 0;

}

**PUSTA LISTA Z GŁOWĄ**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \*next;

};

int main()

{

struct element \* lista = malloc(sizeof(struct element));

lista->next = NULL;

return 0;

}

**DODANIE NA POCZĄTEK – LISTA BEZ GŁOWY**  
(bez uwzględnienia sytuacji kiedy bazowa lista jest pusta)

1. Rezerwacja pamięci na nowy element
2. Ustawienia wartości pola i na nowym elemencie z punktu 1
3. Pole next nowego elementu z punktu 1 jest ustawiane jako adres pierwszej elementu początkowej listy
4. Należy zmienić wskaźnik całej „listy” wskazując jako adres „nowy” element z punktu 1.

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \* next;

};

int main()

{

struct element \* lista = NULL; **//tworzymy "pustą listę"**

**//dodajemy pierwszy element na listę (tu pojęcie koniec czy początek nie ma większego sensu)**

struct element \* wsk = malloc(sizeof(struct element)); **// rezerwujemy miejsce w pamięci na jeden z elementów**

wsk->i=-4;

wsk->next=NULL; **//bo nic więcej nie ma na liście**

lista=wsk; **// ustawiamy wskaznik początku listy**

**// dodajemy nowy element na początek**

struct element \* wsk2 = malloc(sizeof(struct element)); **// rezerwujemy miejsce w pamięci na jeden z elementów**

wsk2->i=7;

wsk2->next=lista; **// następnik nowego elementu to bieżacy wskaznik listy**

lista=wsk2; **// ustawiamy wskaznik nowego elementy jako początek listy**

**// dodajemy jeszcze jeden nowy element na początek**

struct element \* wsk3 = malloc(sizeof(struct element)); **// rezerwujemy miejsce w pamięci na jeden z elementów**

wsk3->i=5;

wsk3->next=lista; **// następnik nowego elementu to bieżacy wskaznik listy**

lista=wsk3; **// ustawiamy wskaznik nowego elementy jako początek listy**

**//tu już mamy listę jak na rysunku - pierwszy eleemnt yo 5, drugi to 7, trzeci to i 4**

**// teraz dodajemy 11 na początek**

struct element \* wsk4 = malloc(sizeof(struct element)); **// rezerwujemy miejsce w pamięci na jeden z elementów**

wsk4->i=11;

wsk4->next=lista; **// następnik nowego elementu to bieżacy wskaznik listy**

lista=wsk4; **// ustawiamy wskaznik nowego elementy jako początek listy**

return 0;

}

**DODAWANIE NA POCZĄTEK LISTY BEZ GŁOWY – GDY LISTA JEST JUŻ STWORZONA**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \* next;

};

int main()

{

struct element \* lista = malloc(sizeof (struct element));

lista->i = 5;

lista->next = malloc(sizeof (struct element));

lista->next->i = 7;

lista->next->next = malloc(sizeof (struct element));

lista->next->next->i = -4;

lista->next->next->next = NULL;

struct element \* temp = lista;

while (temp != NULL)

{

printf("%d\n", temp->i);

temp = temp->next;

}

printf("dodanie\n");

struct element \* wsk = malloc(sizeof (struct element));

wsk->i = 11;

wsk->next = lista;

lista = wsk;

temp = lista;

while (temp != NULL)

{

printf("%d\n", temp->i);

temp = temp->next;

}

return 0;

}

**DODAWANIE NA POCZĄTEK LISTY BEZ GŁOWY – ROZŁOŻONE NA FUNKCJE**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \* next;

};

struct element\* dodaj(struct element\*Lista, int a)

{

struct element \* wsk = malloc(sizeof(struct element));

wsk->i=a;

wsk->next=Lista;

return wsk;

}

struct element \* utworz()

{

return NULL;

}

int main()

{

struct element\* l1 = utworz();

l1 = dodaj(l1,2);

l1 = dodaj(l1,4);

l1 = dodaj(l1,7);

l1 = dodaj(l1,-2);

struct element \* wsk = l1;

while(wsk!=NULL)

{

printf("%d\n",wsk->i);

wsk=wsk->next;

}

return 0;

}

**DODAWANIE NA POCZĄTEK LISTA Z GŁOWĄ**  
(przy założeniu że bieżąca lista nie jest pusta)

1. Rezerwujemy pamięć na nowy element
2. Ustawiamy wartość jako pole i
3. Pole next ustawiamy jako to co znajduje się w „głowie” w polu next
4. Modyfikujemy pole next w „głowie” ustawiając je jako wskaźnik na nowy element z punktu 1

**DODAWANIE NA POCZĄTEK LISTA Z GŁOWĄ – KOD**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \* next;

};

int main()

{

**//tworzymy głowę**

struct element \* lista = malloc(sizeof(struct element));

**//pola i glowy nie ustawiamy**

lista->next= NULL; **//następnik glowy jest na NULL, mamy zatem w tym miejscu pustą listę z głową**

**//dodajemy pierwszy element na listę - w liscie z głową musimy go postawić po głowie**

struct element \* wsk = malloc(sizeof(struct element)); **// rezerwujemy miejsce w pamięci na jeden z elementów**

wsk->i=-4;

wsk->next=lista->next; **//tutaj teoretycznie można ustawić na NULL, ale zgodnie z konwencją trzeba ustawić pole next głowy**

lista->next=wsk; **// ustawiamy nastepnik glowy jako wskaznik na nowy element**

**// dodajemy nowy element na początek**

struct element \* wsk2 = malloc(sizeof(struct element)); **// rezerwujemy miejsce w pamięci na jeden z elementów**

wsk2->i=7;

wsk2->next=lista->next; **// następnik nowego elementu to nastepnik "głowy"**

lista->next=wsk2; **// ustawiamy następnik głowy jako wskaźnik na nowy element**

**// dodajemy jeszcze jeden nowy element na początek**

struct element \* wsk3 = malloc(sizeof(struct element)); **// rezerwujemy miejsce w pamięci na jeden z elementów**

wsk3->i=5;

wsk3->next=lista->next; **// następnik nowego elementu to nastepnik "głowy"**

lista->next=wsk3; **// ustawiamy następnik głowy jako wskaźnik na nowy element**

**//tu już mamy listę jak na rysunku - "głowa", pierwszy eleemnt to 5, drugi to 7, trzeci to i 4**

**// teraz dodajemy 11 na początek**

struct element \* wsk4 = malloc(sizeof(struct element)); **// rezerwujemy miejsce w pamięci na jeden z elementów**

wsk4->i=11;

wsk4->next=lista->next; **// następnik nowego elementu to nastepnik "głowy"**

lista->next=wsk4; **// ustawiamy następnik głowy jako wskaźnik na nowy element**

return 0;

}

**DODAWANIE NA POCZĄTEK LISTA Z GŁOWĄ – KOD PODZIELONY NA FUNKCJE**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \* next;

};

struct element \* utworz()

{

struct element \* wskaznik = malloc(sizeof(struct element));

wskaznik->next=NULL;

return wskaznik;

}

void dodaj(struct element\*Lista, int a)

{

struct element \* wsk = malloc(sizeof(struct element));

wsk->i=a;

wsk->next=Lista->next;

Lista->next=wsk;

}

int main()

{

struct element\* l1 = utworz();

dodaj(l1,2);

dodaj(l1,4);

dodaj(l1,-8);

dodaj(l1,-22);

struct element \* wsk = l1;

while(wsk->next!=NULL)

{

wsk=wsk->next;

printf("%d\n",wsk->i);

}

return 0;

}

**DODANIE NA KONIEC – LISTA BEZ GŁOWY**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \* next;

};

struct element \* dodaj\_na\_koniec(struct element \* Lista, int a)

{

struct element \* wsk;

wsk = Lista;

if (Lista == NULL)

{

Lista = malloc(sizeof(struct element));

Lista->i = a;

Lista->next = NULL;

}

else

{

while (wsk->next != NULL)

{

wsk = wsk->next;

}

wsk->next = malloc(sizeof(struct element));

wsk->next->i = a;

wsk->next->next = NULL;

}

return Lista;

}

void wyswietl(struct element \* Lista)

{

struct element \* wsk;

wsk = Lista;

if (wsk == NULL)

{

printf("Lista jest pusta\n");

return;

}

while (wsk != NULL)

{

printf("%d\n", wsk->i);

wsk = wsk->next;

}

printf("---\n");

}

int main()

{

// przypadek pusty

struct element \* Lista = NULL;

wyswietl(Lista);

Lista = dodaj\_na\_koniec(Lista, 5);

wyswietl(Lista);

// przypadek listy dwuelementowej

struct element \* lista2 = malloc(sizeof(struct element));

lista2->i = 12;

lista2->next = malloc(sizeof(struct element));

lista2->next->i = -8;

lista2->next->next = NULL;

wyswietl(lista2);

lista2 = dodaj\_na\_koniec(lista2, 3);

wyswietl(lista2);

return 0;

}

**DODANIE NA KONIEC – LISTA Z GŁOWĄ**

#include <stdio.h>

#include <stdlib.h>

struct element

{

int i;

struct element \* next;

};

void dodaj\_na\_koniec(struct element \* Lista, int a)

{

struct element \* wsk;

wsk = Lista;

while (wsk->next != NULL)

{

wsk = wsk->next;

}

wsk->next = malloc(sizeof(struct element));

wsk->next->i = a;

wsk->next->next = NULL;

}

void wyswietl(struct element \* Lista)

{

struct element \* wsk;

wsk = Lista->next;

if (wsk == NULL)

{

printf("Lista jest pusta\n");

return;

}

while (wsk != NULL)

{

printf("%d\n", wsk->i);

wsk = wsk->next;

}

printf("---\n");

}

int main()

{

// przypadek pusty

struct element \* Lista1 = malloc(sizeof(struct element));

Lista1->next = NULL;

wyswietl(Lista1);

dodaj\_na\_koniec(Lista1, 5);

wyswietl(Lista1);

// przypadek dwuelementowy

struct element \* Lista2 = malloc(sizeof(struct element));

Lista2->next = malloc(sizeof(struct element));

Lista2->next->i = 12;

Lista2->next->next = malloc(sizeof(struct element));

Lista2->next->next->i = -8;

Lista2->next->next->next = NULL;

wyswietl(Lista2);

dodaj\_na\_koniec(Lista2, 3);

wyswietl(Lista2);

return 0;

}

**USUNIĘCIE ELEMENTU**

free(wsk);