Algorytmy i struktury danych

Projekt

P03 Kamil Bieniek

Kamil Bieniek

Inżynieria i analiza danych, 1. rok, grupa 1.

Lista jednokierunkowa

Lista jednokierunkowa jest sekwencyjną strukturą danych, która składa się z ciągu elementów tego samego typu. Dostęp do elementów listy jest sekwencyjny – tzn. z danego elementu listy możemy przejść do elementu następnego

Naszym zadanie jest pokazanie możliwości operacji na liście jednokierunkowej wraz z napisanie własnej biblioteki

**Użyte biblioteki:**

|  |  |
| --- | --- |
| **iostream** | Biblioteka we-wyjścia. Deklaruje obiekty, które kontrolują odczytywanie ze strumieni standardowych i zapisywanie ich w tych strumieniach. Jest to często jedyny nagłówek potrzebny do wprowadzania danych i danych wyjściowych. |
| **Woid.h** | Biblioteka zawierająca funkcje do obsługi listy jednokierunkowej |
| **windows.h** | Zawiera dosłoweni wszystko , co będzie nam potrzebne w kursie WinAP i będziemy go zawsze dołączać kiedy zechcemy napisać cokolwiek pod windowsa |
| **iomanip** | Dostarcza czas w strukturze czasu i ciągu format do użycia. Umożliwia wygodne zaokrąglanie ciągów za pomocą operatorów wstawiania i wyodrębniania |
| **cstdlib** | Definiuje manipulatory z których każdy ma jeden argument |

**Utworzone funkcje :**

l\_push front ( int x ) - dodaje liczbę na początku listy, przekazuje jaki element mamy dodać

l\_push\_next\_to(int x, int v) - dodaje liczbę za podaną liczbę przez użytkownika i przekazuje za którą liczbę mamy dodać i jaką .

l\_push\_back ( int x) - dodaje liczbę na koniec listy , przekazuje jaki element mamy dodać .

l\_pop\_front() - usuwa liczbę z początku listy .

l\_pop\_next\_to(int v) – usuwa liczbę z podanego miejsca , przekazuje który element od początku listy mamy usunąć.

l\_pop\_back ( ) – usuwa element na końcu listy.

szukaj (int x) – przeszukuje czy podana liczba znajduje się na liście i ile razy występuje, przekazuje jaką liczbę ma poszukać

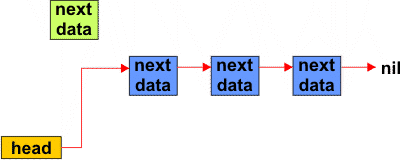
l\_size ( ) – zlicza i pokazuje ile jest elementów na liście

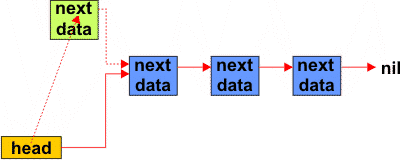
koniec() – kończy program .

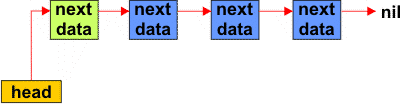
wyswietl() – wyświetla aktualną listę

woid() – deklaruje liste wstawiając wartość NULL w pierwszą pozycje

**Schemat blokowy: l\_push\_front**



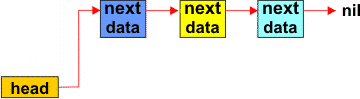


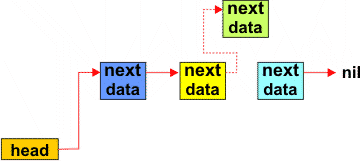


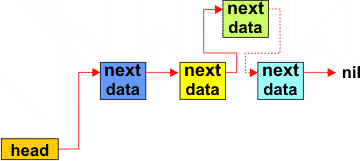
**Pseudokod**

|  |  |
| --- | --- |
| K01 | Utwórz nowy element listy |
| K02 | p <- adres nowego elementu |
| K03 | (p -> liczba) <- podana wartość |
| K04 | (p-> next ) <- head |
| K05 | head <- p |

**Schemat blokowy: l\_push\_next\_to**







p-odnośnik do listy

e – element listy

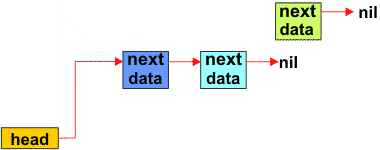
x – podana wartość użytkownika

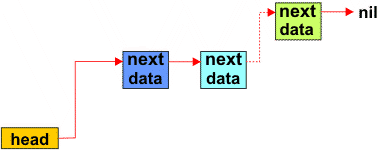
x – podana pozycja

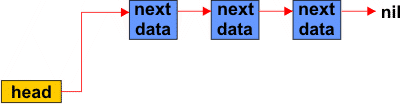
**Pseudokod**

|  |  |
| --- | --- |
| K01 | jeżeli p->liczba <>( V) |
| K02 | e <- adres nowego elementu |
| K03 | e->next = p->next |
| K04 | e->liczba = ( X) |
| K05 | p->next = e |

**Schemat blokowy: l\_push\_back**







p-odnośnik do listy

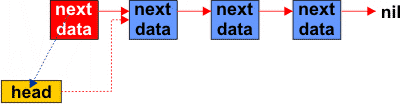
e – element listy

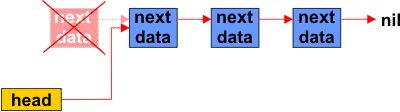
x – podana wartość użytkownika

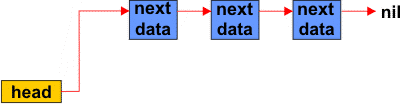
**Pseudokod**

|  |  |
| --- | --- |
| K01 | Utwórz nowy element listy |
| K02 | e <- adres nowego elementu |
| K03 | (e-> liczba) <- x |
| K04 | p = head |
| K05 | Dopóki(p-> next) rób p = p->next |
| K06 | p = p->next |
| K07 | p->next = e |

**Schemat blokowy: l\_pop\_front**





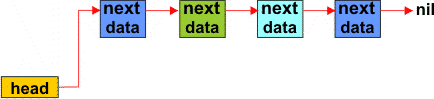


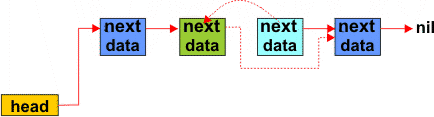
p-odnośnik do listy

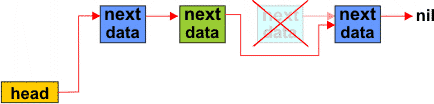
**Pseudokod**

|  |  |
| --- | --- |
| K01 | p<-head |
| K02 | Jeśli (p) |
| K03 | head=l->next |

**Schemat blokowy: l\_pop\_next\_to**







p - odnośnik do listy

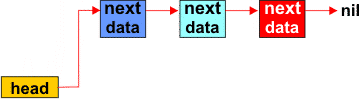
e - element listy

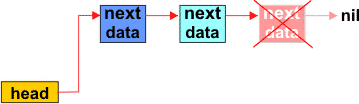
x - podana wartość użytkownika

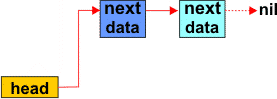
**Pseudokod**

|  |  |
| --- | --- |
| K01 | Dopóki(x > 0) |
| K02 | { p przypisz e |
| K03 | odjąć 1 |
| K04 | e = e->next } |
| K05 | p->next = e->next |

**Schemat blokowy: l\_pop\_back**





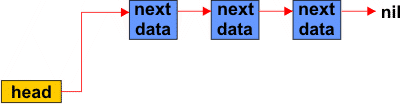


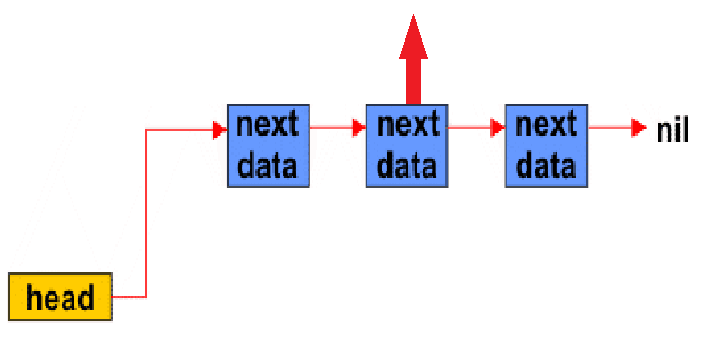
p - odnośnik do listy

**Pseudokod**

|  |  |
| --- | --- |
| K01 | jeśli( p -> next) |
| K02 | Dopóki(p->next->next ) |
| K03 | Wykonuj p = p->next |
| K04 | Usuń p |
| K05 | l->next = NULL |
| K06 | W przeciwnym wypadku |
| K07 | Usuń l |
| K08 | Head <- NULL |

**Schemat blokowy: szukaj**



****

p - odnośnik do listy

x - podana wartość użytkownika

**Pseudokod**

|  |  |
| --- | --- |
| K01 | Utwórz zmienną i<-0 |
| K02 | Jeśli(p->liczba <> x) |
| K03 | i dodaj 1 |
| K04 | Wypisz i |

**Screen z programu :**

