**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Wprowadzenie do Informatyki | **Zadanie** | 1 |
| **Autor** | Mateusz Jasiński | **Grupa** | WCY20IY2S1 |
| **Temat** | Struktury dynamiczne - listy | | |

1. Treść

Ewidencja książek/samochodów w postaci listy jednokierunkowej realizująca operacje:

-Dodaj obiekt,

-Usuń obiekt,

-Znajdź obiekt,

-Wylistuj zapisane obiekty.

* 1. Metoda realizacji

Pobrać wartość polecenie, która określa która z funkcji ma zostać wywołana. W zależności od wywołanej funkcji:  
-Dodaj obiekt – pobrać dane i dodać je na początek listy  
-Usuń obiekt – pobrać dane i usunąć element listy, który się z nimi pokrywa  
-Znajdź obiekt – pobrać dane i wyświetlić obiekt, który się z nimi pokrywa  
-Wylistuj zapisane elementy – wyświetlić wszystkie elementy listy  
Program kończy działanie gdy polecenie otrzyma wartość 0, w przeciwnym wypadku wraca do pobrania polecenia i kontynuuje działanie.

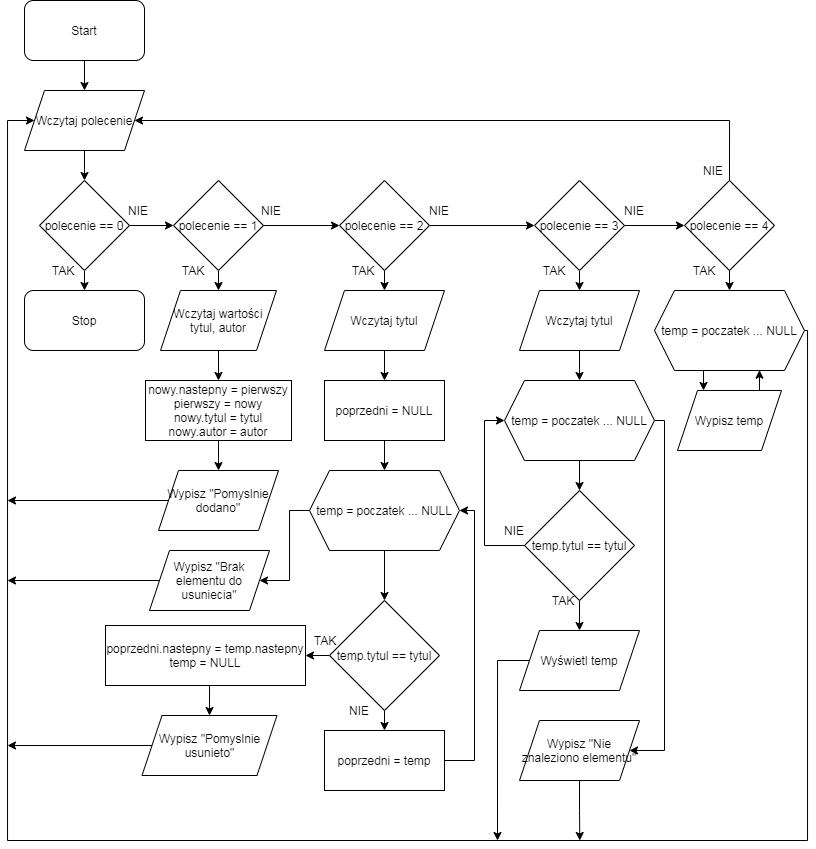
* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe

Wartość polecenie, w zależności od wywołanej funkcji: dane\_do\_dodania, dane\_do\_usuniecia, dane\_do\_wyszukania – wprowadzone z klawiatury

* + 1. Dane wyjściowe

W zależności od funkcji: komunikat o dodaniu, komunikat o usunięciu, znaleziony obiekt listy, wszystkie elementy listy – wyprowadzone na ekran

1. Realizacja
   1. Algorytm



* 1. Kod źródłowy

#include <stdio.h>

#include <stdlib.h>

typedef struct ksiazka{

char tytul[40];

char autor[40];

struct ksiazka \*nastepna;

} ksiazka;

int gui() {

system("cls");

printf("Ewidencja ksiazek\n"

"----------------------------------------------\n"

"[0] Wyjscie z programu\n"

"[1] Dodaj ksiazke\n"

"[2] Usun ksiazke\n"

"[3] Szukaj ksiazki po tytule\n"

"[4] Wyswietl liste ksiazek\n"

"----------------------------------------------\n\n"

"Wybrana komenda: ");

}

dodaj(ksiazka \*\*poczatek) {

ksiazka \*nowa;

nowa=(ksiazka\*)malloc(sizeof(ksiazka));

nowa->nastepna=\*poczatek;

\*poczatek=nowa;

printf("Dodawanie ksiazki\n"

"----------------------------------------------\n"

"Podaj tytul: ");

fflush(stdin);

gets(nowa->tytul);

printf("Podaj imie i nazwisko autora: ");

gets(nowa->autor);

fflush(stdin);

printf("Pomysle dodano!\nKliknij aby kontynuowac.");

getch();

}

int usun2(ksiazka \*poczatek, char tytul[]) {

ksiazka \*pop=poczatek;

while(poczatek) {

if(!strcmp(poczatek->tytul,tytul)) {

pop->nastepna=poczatek->nastepna;

free(poczatek);

return 1;

} else {

pop=poczatek;

poczatek=poczatek->nastepna;

}

}

return 0;

}

usun(ksiazka \*\*poczatek) {

if(\*poczatek!=NULL) {

int cos;

char tytul[40];

ksiazka \*do\_usuniecia;

do\_usuniecia=\*poczatek;

printf("Podaj tytul ksiazki do usuniecia: ");

fflush(stdin);

gets(tytul);

fflush(stdin);

if(!strcmp(do\_usuniecia->tytul,tytul)) {

\*poczatek=do\_usuniecia->nastepna;

free(do\_usuniecia);

cos=1;

} else

cos=usun2(\*poczatek,tytul);

if(cos)

printf("Pomysle usunieto!\n");

else

printf("Nie znaleziono ksiazki o danym tytule!\n");

} else

printf("Lista jest pusta, dodaj cos!\n");

printf("\nKliknij aby kontynuowac.");

getch();

}

szukaj(ksiazka \*poczatek) {

ksiazka \*i;

int cos=0, poz;

char tytul[40];

if(poczatek==NULL) {

printf("Lista jest pusta, nie ma czego szukac.\n");

cos=1;

}

else {

printf("Podaj tytul ksiazki, ktorej szukasz: ");

fflush(stdin);

gets(tytul);

fflush(stdin);

for(i=poczatek, poz=1; i!=NULL; i=i->nastepna, poz++) {

if(!strcmp(i->tytul,tytul)) {

printf("Lp.%d\t", poz);

printf("Tytul: %s\t", i->tytul);

printf("Autor: %s\t", i->autor);

cos=1;

break;

}

}

}

if(!cos)

printf("Nie znaleziono ksiazki o danym tytule.\n");

printf("\nKliknij aby kontynuowac.");

getch();

}

wyswietl(ksiazka \*poczatek) {

ksiazka \*i;

int poz=1;

if(poczatek==NULL)

printf("Lista jest pusta, dodaj cos!\n");

else

for(i=poczatek; i!=NULL; i=i->nastepna, poz++) {

printf("Lp.%d\t", poz);

printf("Tytul: %s\t", i->tytul);

printf("Autor: %s\t", i->autor);

printf("\n");

}

printf("\nKliknij aby kontynuowac.");

getch();

}

main() {

int polecenie;

ksiazka \*pierwsza = NULL;

do {

gui();

scanf("%d", &polecenie);

system("cls");

switch(polecenie) {

case 0:

printf("Zakonczono dzialanie programu.");

break;

case 1:

dodaj(&pierwsza);

break;

case 2:

usun(&pierwsza);

break;

case 3:

szukaj(pierwsza);

break;

case 4:

wyswietl(pierwsza);

break;

}

} while(polecenie);

if(pierwsza==NULL) //usuwanie elementow listy ktore zostaly po dzialaniu programu

free(pierwsza);

else

while(1) {

if(pierwsza->nastepna==NULL) {

free(pierwsza);

break;

} else {

ksiazka \*pop=pierwsza;

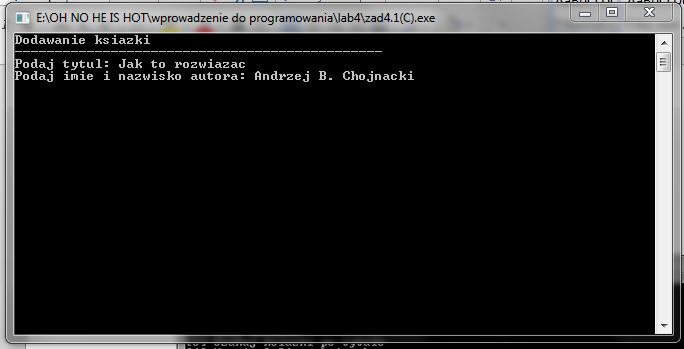
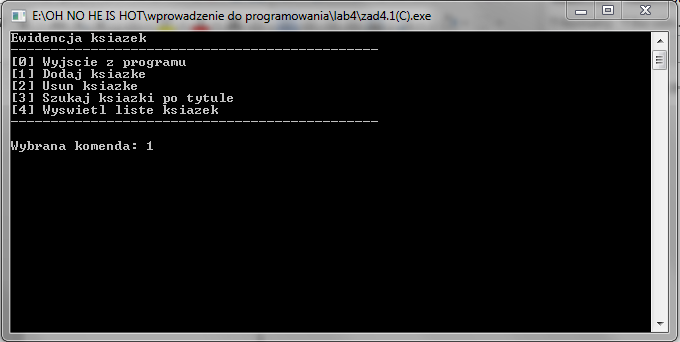
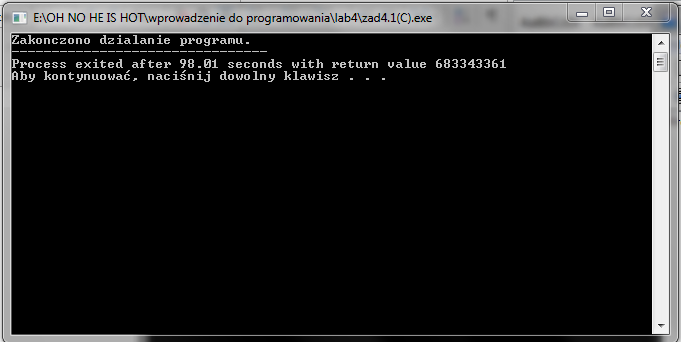
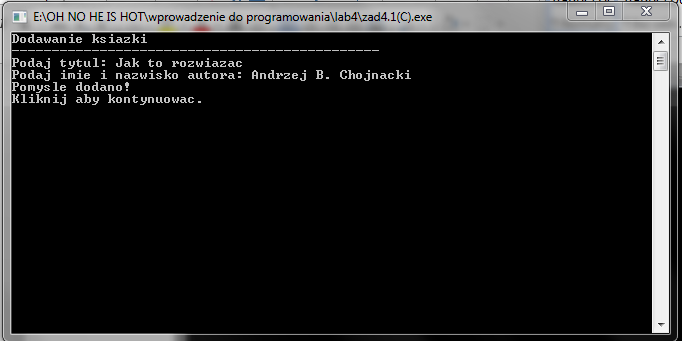
pierwsza=pierwsza->nastepna;

free(pop);

}

}

}

* 1. Dane wejściowe
  2. Dane wyjściowe

1. Wnioski

Złożoność obliczeniowa algorytmu:

O(n) = n + n + 4n + n = 7n

Złożoność obliczeniowa jest liniowa, gdyż jest ona uzależniona od liczby poleceń. Najbardziej złożone obliczeniowo polecenie, w tym programie, to dodawanie.