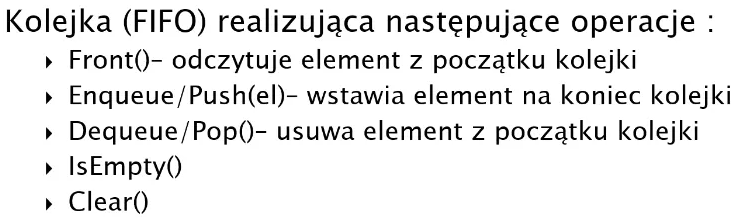
**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Wprowadzenie do Informatyki | **Zadanie** | 1 |
| **Autor** | Mateusz Jasiński | **Grupa** | WCY20IY2S1 |
| **Temat** | Struktury dynamiczne – kolejki i stosy | | |

1. Treść



* 1. Metoda realizacji

W zależności od wybranego polecenia:

-Odczytać element, który jest ostatnim elementem kolejki,  
-Dodać element na początek kolejki,  
-Usunąć ostatni element kolejki,  
-Sprawdzić czy kolejka jest pusta,  
-Usunąć wszystkie elementy kolejki.

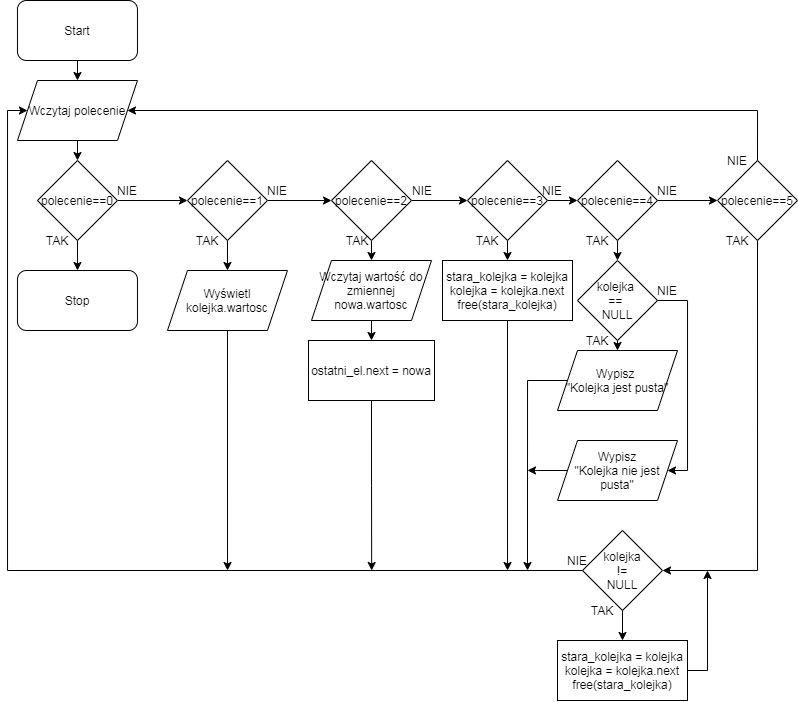
* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe

Dane obiektów – wprowadzone z klawiatury

* + 1. Dane wyjściowe

Dane zapisanych obiektów – wyprowadzone na ekran

1. Realizacja
   1. Algorytm



* 1. Kod źródłowy

#include<stdio.h>

#include<stdlib.h>

typedef struct fifo {

int wartosc;

struct fifo \*nastepny;

} fifo;

int gui() {

system("cls");

printf("Kolejka FIFO\n"

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

"[0] Wyjscie z programu\n"

"[1] Front()\n"

"[2] Enqueue(el)\n"

"[3] Dequeue()\n"

"[4] IsEmpty()\n"

"[5] Clear()\n"

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

"Wybrana komenda: ");

}

int IsEmpty(fifo \*kolejka) {

if(kolejka==NULL)

return 1;

else

return 0;

}

Front(fifo \*kolejka) {

if(IsEmpty(kolejka))

printf("Kolejka jest pusta!\n");

else

printf("%d\n",kolejka->wartosc);

printf("\nKliknij aby kontynuowac.");

getch();

}

Enqueue(fifo \*\*kolejka) {

fifo \*nowy;

nowy=(fifo\*)malloc(sizeof(fifo));

nowy->nastepny=NULL;

printf("Podaj liczbe do wstawienia: ");

scanf("%d", &nowy->wartosc);

if(IsEmpty(\*kolejka))

\*kolejka=nowy;

else {

fifo \*nowy2=\*kolejka;

while(nowy2->nastepny!=NULL)

nowy2=nowy2->nastepny;

nowy2->nastepny=nowy;

}

printf("\nPomyslenie dodano element na koniec kolejki\nKliknij aby kontynuowac.");

getch();

}

Dequeue(fifo \*\*kolejka) {

if(IsEmpty(\*kolejka))

printf("Kolejka jest pusta!\n");

else {

fifo \*temp=\*kolejka;

fifo \*pop=temp;

temp=temp->nastepny;

free(pop);

\*kolejka=temp;

printf("Pomyslnie usunieto element na poczatku kolejki");

}

printf("\nKliknij aby kontynuowac.");

getch();

}

Clear(fifo \*\*kolejka) {

fifo \*temp=\*kolejka;

while(!IsEmpty(temp)) {

fifo \*pop=temp;

temp=temp->nastepny;

free(pop);

}

\*kolejka=NULL;

printf("Pomyslnie wyczyszczono kolejke\nKliknij aby kontynuowac.");

getch();

}

main() {

fifo \*kolejka=NULL;

int polecenie;

do {

gui();

scanf("%d", &polecenie);

system("cls");

switch(polecenie) {

case 0:

printf("Zakonczono dzialanie programu.");

break;

case 1:

Front(kolejka);

break;

case 2:

Enqueue(&kolejka);

break;

case 3:

Dequeue(&kolejka);

break;

case 4:

if(IsEmpty(kolejka))

printf("Kolejka jest pusta.\n");

else

printf("Kolejka nie jest pusta.\n");

printf("\nKliknij aby kontynuowac.");

getch();

break;

case 5:

Clear(&kolejka);

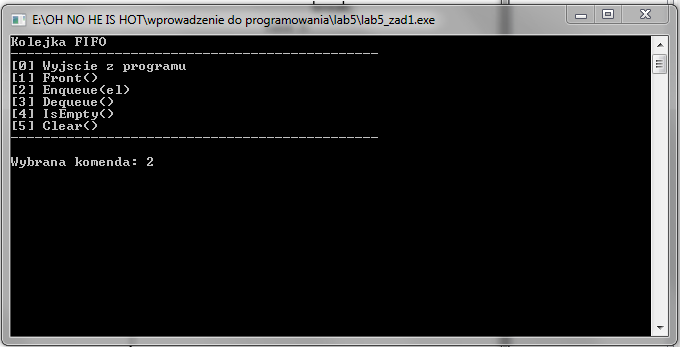
break;

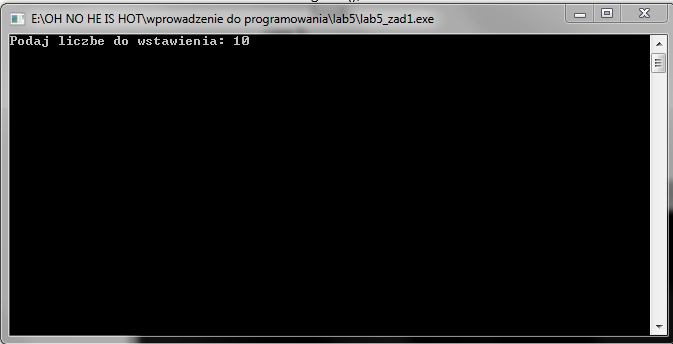
}

} while(polecenie);

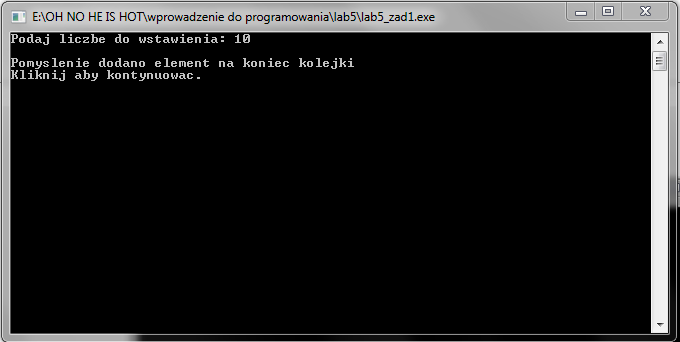
}

* 1. Dane wejściowe





* 1. Dane wyjściowe



1. Wnioski

O(n) – złożoność obliczeniowa zależy liniowo od liczby wprowadzonych elementów.