Politechnika Rzeszowska
Wydział Matematyki i Fizyki Stosowanej
Inżynieria i analiza danych

# PROJEKT NR III

## Algorytmy i struktury danych

Pracę wykonała:
Aleksandra Słaby

Rzeszów, 10 stycznia 2022

# 1 Wstęp

## 1.1 Temat projektu

Dokonaj implementacji struktury danych typu **lista jednokierunkowa** wraz z wszelkimi potrzebnymi operacjami charakterystycznymi dla tej struktury (inicjowanie struktury, dodawanie i odejmowanie elementów, wyświetlanie elementów, zliczanie elementów, wyszukiwanie zadanego elementu itp.)
Należy:

1. Przyjąć, że podstawowym typem danych przechowywanych w elemencie struktury będzie struktura z jednym polem typu int;
2. W funkcji main przedstawić możliwości napisanej przez siebie biblioteki;
3. Kod powinien być opatrzony stosownymi komentarzami.

Program miał zostać napisane w środowisku Code::Blocks IDE w języku C/C++.

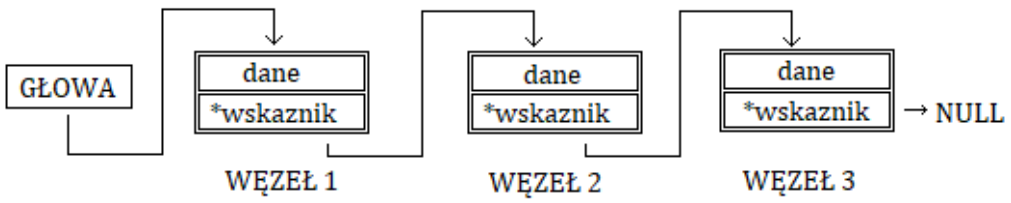
## 1.2 Cel projektu

Celem projektu było poznanie i dokonanie właściwej implementacji struktury danych typu **lista jednokierunkowa**, a następnie zaprezentowanie specyficznych dla niej operacji. Działanie na strukturach dynamicznych jest niezwykle przydatną umiejętnością dla analityka danych.

# 2 Zagadnienia teoretyczne

## 2.1 Lista jednokierunkowa - podstawowe informacje

**Lista jednokierunkowa** to struktura danych z elementami ułożonymi w liniowym porządku ustalonym przez wskaźniki pozwalająca na grupowanie dowolnej ilości danych, ograniczonej jedynie przez ilość dostępnej pamięci. Jej użycie wymaga zarezerwowania w pamięci miejsca na informacje wskaźnikowe, jednakże nie wpływa to na fakt, iż lista jest strukturą bardzo oszczędną pamięciowo. **Węzeł**, czyli każdy element listy, zawiera dwa pola - jedno na konkretne dane i drugie zawierające wskaźnik na następny element. Pierwszy element listy nazywany jest **głową**, a ostatni **ogonem** (w wersji anglojęzycznej dosłowne tłumaczenie: head, tail). Wskaźnik ostatniego elementu wskazywać powinien na NULL.
W odróżnieniu od listy dwukierunkowej użytkownik może przesuwać się po elementach wyłącznie w jednym kierunku - od początku do końca listy.




Rysunek 1: Lista jednokierunkowa

Powyższy schemat obrazuje trójelementową listę jednokierunkową. Danymi każdego elementu mogą być zmienna lub kilka zmiennych. Tworząc bardziej rozbudowane listy możemy używać zmiennych różnego typu, np. **int** rok, **string** wydarzenie.

## 2.2 Wady i zalety listy jednokierunkowej

**Zalety:**

1. Oszczędność pamięci - dynamiczne przydzielanie i zwalnianie pamięci na poszczególne węzły;
2. Wysoka elastyczność rozmiaru listy;
3. Możliwość wykonywania efektywnych operacji na danych;

**Wady:**

1. Problematyczne sortowanie listy;
2. Utrudniona ocena wielkości listy;
3. W odróżnieniu do list dwukierunkowych przez swoją specyfikę dojście do przedostatniego elementu wymaga przejścia przez prawie całą listę;

# 3 Działania na strukturze

## 3.1 Inicjowanie listy

Na początku musimy utworzyć strukturę elementu listy i zdefiniować podział na dane i wskaźnik do kolejnego elementu. W moim przypadku lista USOS zwierała w sobie indeksy poszczególnych studentów, które zawierały się w zmiennej **int** indeks. USOS() jest konstruktorem, jednakże nie będziemy przekazywać do niego póki co żadnych danych, gdyż wczytane zostaną one później z pliku.

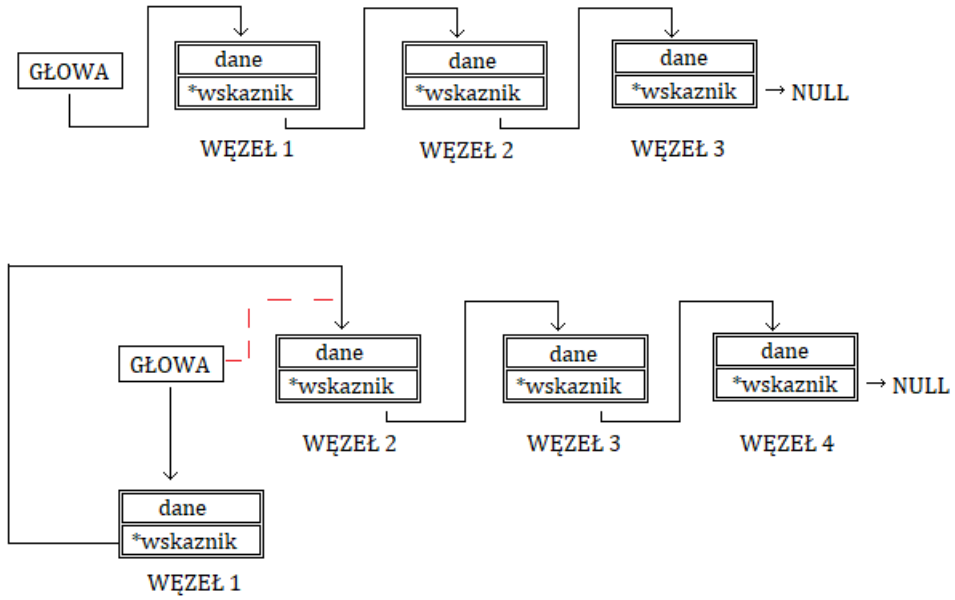
```
struct USOS
{
   int indeks;
   USOS *nastepny;

   USOS();


USOS(int indeks)
   {
       this->indeks = indeks;
       nastepny = NULL;
   }
 };
```

## 3.2 Dodawanie elementów do listy

### 3.2.1 Dodawanie elementów na początku

Żeby dodać węzeł na początek listy należy utworzyć nowy element i przypisać mu wartość, a następnie wskaźnik **nastepny* ustawić na *glowa*, by wskazywał on na poprzedzającą go głowę.

```
 void na_poczatek(USOS **glowa, int nowy_indeks)
{

    USOS *nowa = new USOS(nowy_indeks); //nowy element

    nowa->nastepny = (*glowa);

    (*glowa) = nowa; //dodaj na poczatek (glowa)
}
```




Rysunek 2: Schemat dodawania elementu na początek listy

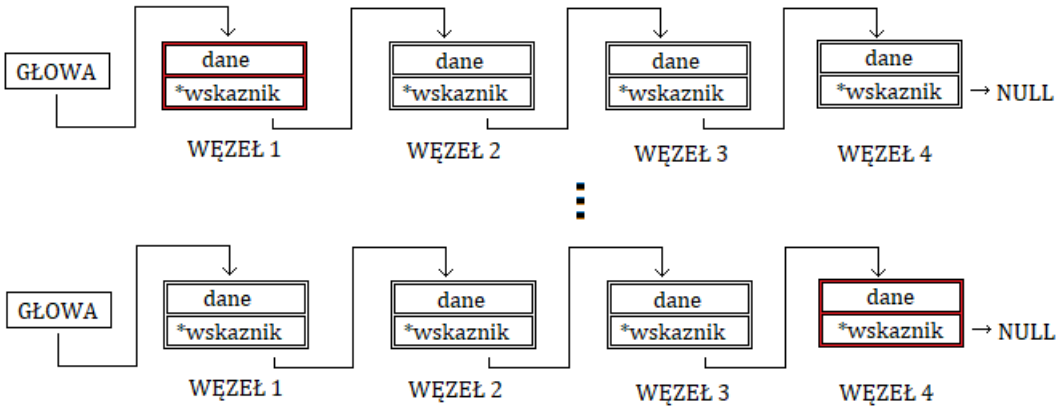
Na powyższym zdjęciu widoczna jest lista przed i po dodaniu węzła. Czerwoną przerywaną linią zaznaczony był poprzedni wskaźnik, który przypisano nowemu elementowi.

### 3.2.2 Dodawanie elementu na koniec listy

Podczas dodawania elementu na koniec listy należy rozważyć przypadek, w którym lista jest pusta oraz standardowy, kiedy ma ona jakieś elementy. W pierwszym wypadku należy ustawić nowy element jako głowę, gdyż będzie ona zarówno pierwszym jak i ostatnim elementem listy. Dla listy niepustej należy przejść po całej liście, aby dostać się na jej koniec i tam dodać element.

```
 void na_koniec(USOS *&glowa, int nowy_indeks)
{
    USOS *nowa = new USOS(nowy_indeks);
    nowa->indeks = nowy_indeks;

    if (glowa==0)
    {
        glowa = nowa;
    }
        else
        {
        USOS *temp = glowa;

            while (temp->nastepny)
            {
                temp = temp->nastepny;
            }

        temp->nastepny = nowa;
        }
}
```

Na poniższym schemacie kolorem czerwonym oznaczany jest nowy element dodany do listy, który przechodząc przez kolejne pozycje listy, dociera na ostatnią pozycję.




Rysunek 3: Schemat dodawania elementu na koniec listy

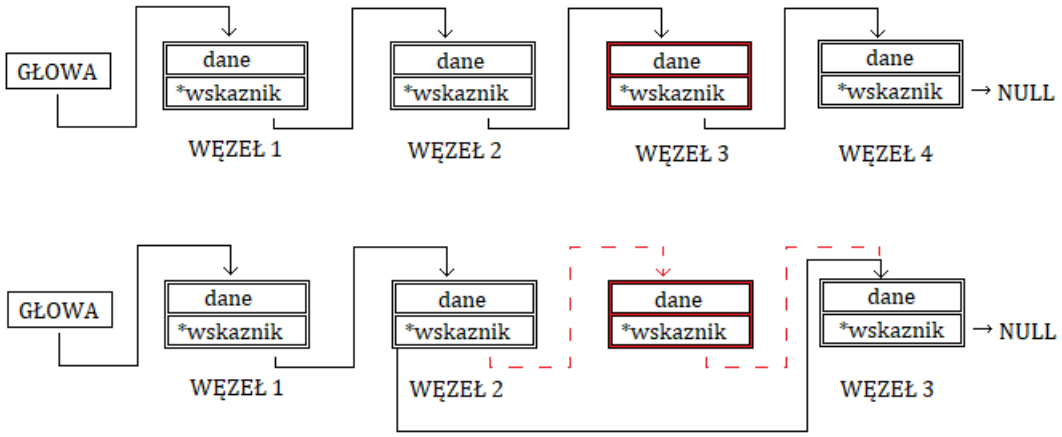
## 3.3 Usuwanie elementu z listy

Usuwanie elementu z listy polega na znalezieniu zadanego przez użytkownika indeksu i usunięcie go z listy. W przypadku pustej listy funkcja bool nie będzie nic zwracać, natomiast gdy posiadamy jakiekolwiek elementy funkcja będzie przeszukiwać listę aż do napotkania zadanego indeksu lub jej końca.

```
bool usuwanie_zadanego(USOS **glowa, int student)
{
    if ((*glowa) == NULL)
    return false;

    USOS *temp = (*glowa), *temp2 = NULL;

        while(temp != NULL && (temp->indeks)!=student)
            {
            temp2 = temp;
            temp = temp->nastepny;
            }

    if (temp == NULL)
    return false;

        else if (temp == (*glowa))
        {
        (*glowa) = (*glowa)->nastepny;
        delete temp;
        }
            else
            {
            temp2->nastepny = temp->nastepny;
            delete temp;
            }
        return true;
}
```

W przypadku kiedy podany indeks nie pojawia się na liście, podczas wywoływania funkcji podany został warunek, który powiadomi użytkownika o błędzie lub poprawnym znalezieniu użytkownika:

```
if (!usuwanie_zadanego(&glowa, indeks))
cout << "Nie odnaleziono studenta z podanym indeksem";
        else
    drukowanie(glowa);
    cout << "Konto USOS z podanym indeksem zostalo usuniete";
```

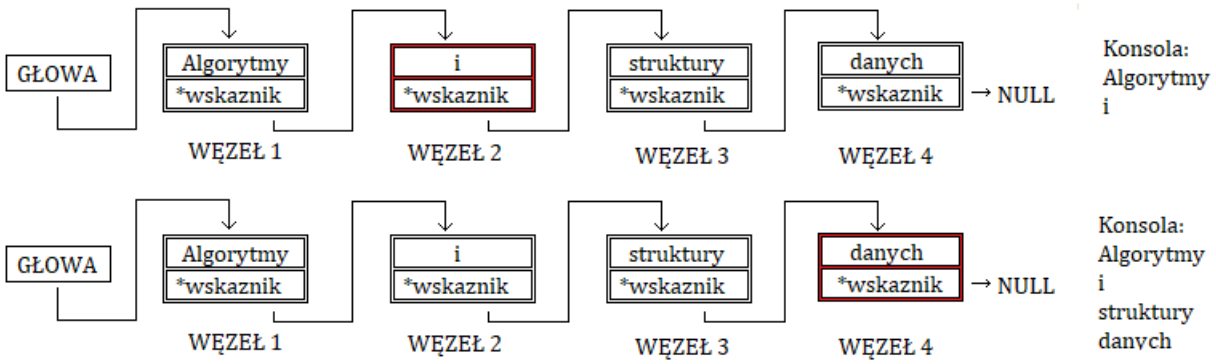



Rysunek 4: Schemat usuwania zadanego elementu

## 3.4 Wyświetlanie listy

Wyświetlenie wszystkich elementów listy jest dosyć prostą operacją, polegającą na przejściu przez całą listę i wypisywanie każdego węzła, poczynając od głowy. Na poniższym schemacie zaprezentowany jest wynik konsoli w momencie, gdy wskaźnik przebiega przez wskazane elementy.

```
void wyswietlanie(USOS *glowa)
        {
    cout << "Lista studentow :"<< endl;

    while(glowa != NULL)
        {
        glowa->wypisz();
        glowa = glowa->nastepny;
        }
    }

    Gdzie:
    void wypisz()
    {
    cout << "Numer indeksu: " << indeks << endl;
    }
```




Rysunek 5: Schemat działania wyświetlania listy

## 3.5 Wyszukiwanie elementu listy

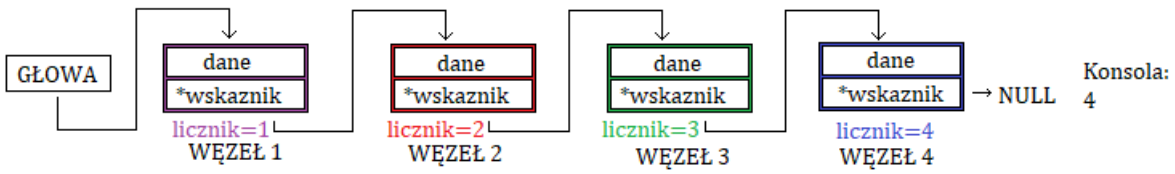
Wyszukiwanie elementu na liście jest jedną z najprostszych operacji, a na jej podstawie działało wcześniejsze usuwanie zadanego elementu. Użytkownik podaje indeks studenta, którego poszukuje, a funkcja przeszukuje każdy z elementów do momentu znalezienia go lub zakończenia się listy.

```
void wyszukiwanie(USOS *glowa, int student)
{
    while(glowa != NULL && (glowa->indeks)!=student)
        {
        glowa = glowa->nastepny;
        }
        if (glowa == NULL)
            cout << "Student o takim indeksie nie istnieje. Sprawdz
                poprawnosc indeksu i sprobuj ponownie. "  << endl;
        else
            glowa->wypisz();
}
```

## 3.6 Zliczanie elementów na liście

Do wykonania operacji zliczania potrzebny jest licznik, który zliczy wszystkie elementy. Począwszy od głowy funkcja powinna przejść przez wszystkie węzły aż do wskaźnika NULL, zwiększając z każdym przejściem licznik.

```
void zliczanie(USOS *glowa)
{
    USOS *temp;
    int licznik = 0;

     temp = glowa;

        while(temp != NULL)
        {
            licznik++;
            temp = temp->nastepny;
        }
    cout<< "Liczba elementow listy: " << licznik << endl;
}
```

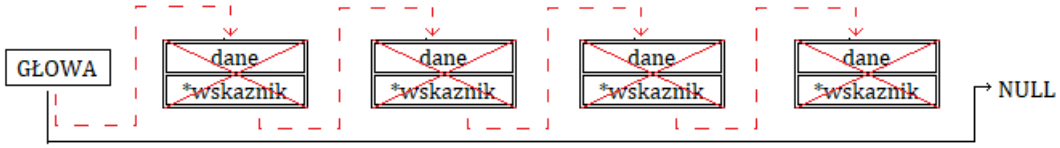



Rysunek 6: Schemat zliczania wszystkich elementu na liście

## 3.7 Resetowanie listy

Jeżeli chcemy rozpocząć pisanie listy od początku, a elementów jest zbyt dużo, by usuwać każdy po kolei, można użyć funkcji, która zresetuje całą listę. Jej działanie jak w większości innych przypadków, oparte jest na przejściu przez całą listę, usuwając kolejne węzły, aż do dojścia do wskaźnika NULL.

```
void reset(USOS *&glowa)
{    USOS *nowa;
    while (glowa != NULL)
    {   nowa = glowa;
        glowa = nowa->nastepny;
        delete nowa;
    }
}
```



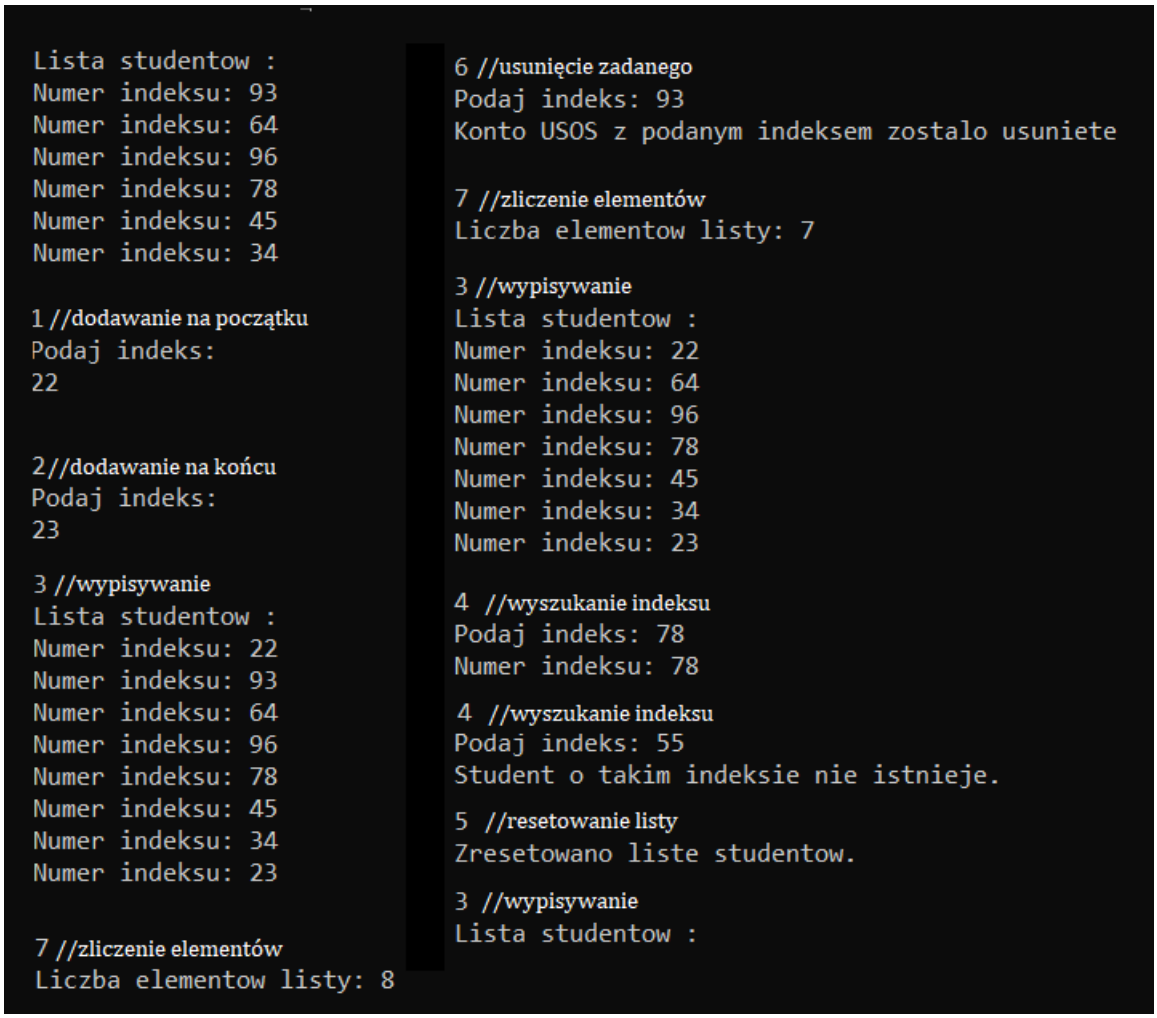

Rysunek 7: Schemat resetowania całej listy

# 4 Test

```
Witaj w systemie USOS! Wybierz, co chcesz zrobic:
'1' jesli chcesz dodac studenta do USOSA na poczatek listy
'2' jesli chcesz dodac studenta na koniec listy
'3' jesli chcesz wypisac liste indeksow znajdujaca sie w USOSie
'4' jesli chcesz wyszukac konkretnego studenta po jego indeksie
'5' jesli chcesz zresetowac cala liste. Operacji nie da sie cofnac
'6' jesli chcesz usunac studenta z USOSA
'7' jesli chcesz poznac ilosc elementow na liscie
'X' jesi chcesz zakonczyc i zapisac program
W razie pytan technicznych prosimy o kontakt na maila 169840@stud.prz.edu.pl
```

Rysunek 8: Widok konsoli z menu wyboru

Po wywołaniu programu użytkownik może sterować listą za pomocą znaków (**char**) wpisywanych przez niego ręcznie. Na początku na liście znajdują się 24 indeksy pobrane z pliku *dane.txt*, jednakże resetując listę możemy zacząć wypełnianie jej od początku. Na poniższym zrzucie ekranu widoczna jest praca programu i przykładowe operacje wykonywane na liście: dodawanie węzła na początku i na końcu listy, wyświetlanie całej listy, usuwanie zadanego elementu, zliczanie elementów, wyszukiwanie konkretnego węzła oraz resetowanie całości. W celu zapisania jej do pliku *wyniki.txt* użytkownik musi odpowiednio zakończyć program.




Rysunek 9: Wyniki testu dla poszczególnych działań

# 5 Bibliografia


1. Wszystkie rysunki zostały wykonane przez autora na bazie `https://upload.wikimedia.org/wikibooks/pl/d/d6/Lista-jednokierunkowa.gif`
2. Wróblewski Piotr, Algorytmy, struktury danych i techniki programowania
3. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C., Wprowadzenie do algorytmów `https://elektronika327.blogspot.com/2016/05/c-lista-jednokierunkowa.html`
4. `https://www.p-programowanie.pl/cpp/lista-jednokierunkowa-c#jak-napisa%C4%87-list%C4%99-jednokierunkow%C4%85`
5. `https://binarnie.pl/lista-jednokierunkowa/`


# 6 Github

`https://github.com/aleksandraslaby/aisd_3`

# 7 Załącznik: Kod programu

```
#include <iostream>
#include <fstream>
#include <cstdlib>

 // LISTA JEDNOKIERUNKOWA
 // ALEKSANDRA SLABY
 // 169840, 1FS-DI

using namespace std;


struct USOS //lista USOS ktorej elementami beda indeksy studentow
    {
    int indeks; //indeks kazdego studenta zawiera sie w zakresie wartosci int
    USOS *nastepny; // wskaznik na nastepny element na liscie

    USOS(); // konstruktor do wywolywania przy tworzeniu listy


 USOS(int indeks)
    {
        this->indeks = indeks;
        nastepny = NULL;
    }

void wypisz() //funkcja wypisujaca element z listy, zostanie uzyta pozniej
    {
    cout << "Numer indeksu: " << indeks << endl;
    }
    };

    //WYSZUKIWANIE ELEMENTOW
void wyszukiwanie(USOS *glowa, int student) //wyszukiwanie konkretnego
    indeksu studenta
{
    while(glowa != NULL && (glowa->indeks)!=student) // petla wykonuje sie
        dopoki nie znajdziemy konta z podanym indeksem lub nie przeszukamy
        calej listy
        {
        glowa = glowa->nastepny;
        }
        if (glowa == NULL) //gdy dojdziemy do konca, a indeksu nie ma to
            wyswietlamy komunikat
            cout << "Student o takim indeksie nie istnieje. Sprawdz
                poprawnosc indeksu i sprobuj ponownie. "  << endl;
        else
            glowa->wypisz();

}

    //WYSWIETLANIE CALEJ LISTY
void wyswietlanie(USOS *glowa) //wyswietlamy liste elementow
    {
    cout << "Lista studentow :"<< endl;

    while(glowa != NULL) //dopoki nie dojdziemy do konca
        {
        glowa->wypisz(); // wypisanie konta
        glowa = glowa->nastepny; // przejscie na kolejny element
        }
    }
```

```
    //USUWANIE PODANEGO ELEMENTU
bool usuwanie_zadanego(USOS **glowa, int student) //usuwanie studenta o
    podanym indeksie
{
    if ((*glowa) == NULL) //najpierw sprawdzamy czy na liscie mamy
        jakiekolwiek elementy
    return false;

    USOS *temp = (*glowa), *temp2 = NULL; //wskazniki do sprawdzania listy

        while(temp != NULL && (temp->indeks)!=student) // dopoki nie
            znajdziemy konta USOS z podanym indeksem lub nie przeszukamy
            calego USOSA
            {
            temp2 = temp;
            temp = temp->nastepny;
            }

    if (temp == NULL) //jesli doszlismy do konca a indeksu nie ma zwracamy
        false
    return false;

        else if (temp == (*glowa))
        {
        (*glowa) = (*glowa)->nastepny;
        delete temp;
        }
            else
            {
            temp2->nastepny = temp->nastepny;
            delete temp;
            }
        return true;
}

    //DODAWANIE ELEMENTU NA KONIEC LISTY
void na_koniec(USOS *&glowa, int nowy_indeks) //dodawanie na koniec
{
    USOS *nowa = new USOS(nowy_indeks);    // tworzy nowy element listy
    nowa->indeks = nowy_indeks;

    if (glowa==0) // sprawdzamy czy to pierwszy element listy
    {
        // jezeli tak to nowy element jest teraz poczatkiem listy
        glowa = nowa;
    }
        else
        {
        // w przeciwnym wypadku wedrujemy na koniec listy
        USOS *temp = glowa;

            while (temp->nastepny) // znajdujemy wskaznik na ostatni element
            {
                temp = temp->nastepny;
            }

        temp->nastepny = nowa;  // ostatni element wskazuje na nasz nowy
            element


        }
}
```

```
    //DODAWANIE ELEMENTU NA POCZATEK LISTY
void na_poczatek(USOS **glowa, int nowy_indeks) //dodawanie na poczatek
{

    USOS *nowa = new USOS(nowy_indeks); //nowy element

    nowa->nastepny = (*glowa);

    (*glowa) = nowa; //dodaj na poczatek (glowa)
}


    //RESETOWANIE CALEJ LISTY
void reset(USOS *&glowa) //usuwamy cala liste
{
    USOS *nowa;
    while (glowa != NULL) //dopoki lista nie bedzie pusta usuwamy elementy
    {
        nowa = glowa;
        glowa = nowa->nastepny;
        delete nowa;
    }
}

    //ZLICZANIE WSZYSTKICH ELEMENTOW
void zliczanie(USOS *glowa)
{
    USOS *temp;
    int licznik = 0; //poczatkowy licznik ustawiamy na zero

     temp = glowa;

        while(temp != NULL) //dopoki nie przejdziemy przez wszystkie elementy
             to zwiekszamy licznik
        {
            licznik++;
            temp = temp->nastepny;
        }

    cout<< "Liczba elementow listy: " << licznik << endl;
}

//FUNKCJA WYPISUJACA DO PLIKU
void drukowanie(USOS *glowa)

{
    ofstream wynik;
       wynik.open("lista.txt");

        while (glowa != NULL) {
                wynik << glowa->indeks<<endl;

                glowa = glowa->nastepny;
        }
}

//ODCZYT CZESCI DANYCH Z PLIKU
void wczytywanie(USOS * & glowa)
{
    USOS * nowa, * ostatni;
    ostatni = nowa = NULL;
```

```
    ifstream plik;
    plik.open("dane.txt");

    int indeksWczytywany;

        while (!plik.eof())
            {
            plik >> indeksWczytywany;

            ostatni = nowa;
            nowa = new USOS(indeksWczytywany);

            nowa -> nastepny = NULL;
            if (ostatni == NULL)
                glowa = nowa;
            else
            ostatni -> nastepny = nowa;

            }
}


    int main()
    {
    char opcja;
    int indeks;
    USOS *glowa = NULL; // glowa listy - wskaznik na pierwszy element

    wczytywanie(glowa);

        cout << "Witaj w systemie USOS! Wybierz, co chcesz zrobic: "<<endl;
        cout << "'1' jesli chcesz dodac studenta do USOSA na poczatek listy"
            <<endl;
        cout << "'2' jesli chcesz dodac studenta na koniec listy"<<endl;
        cout << "'3' jesli chcesz wypisac liste indeksow znajdujaca sie w
            USOSie"<<endl;
        cout << "'4' jesli chcesz wyszukac konkretnego studenta po jego
            indeksie"<<endl;
        cout << "'5' jesli chcesz zresetowac cala liste. Operacji nie da sie
            cofnac"<<endl;
        cout << "'6' jesli chcesz usunac studenta z USOSA"<<endl;
        cout << "'7' jesli chcesz poznac ilosc elementow na liscie"<<endl;
        cout << "'X' jesi chcesz zakonczyc i zapisac program"<<endl;
        cout << "W razie pytan technicznych prosimy o kontakt na maila 169840
            Gstud.prz.edu.pl "<<endl;


while (cin >> opcja)
{
    switch (opcja)
    {
        case '1':
        cout<< "Podaj indeks: "<< endl;
        cin >> indeks;
        na_poczatek(&glowa, indeks);
        drukowanie(glowa);
        break;

            case '2':
            cout<<"Podaj indeks: "<< endl;
            cin >> indeks;
            na_koniec(glowa, indeks);
            drukowanie(glowa);
```

```
            break;

        case '3': wyswietlanie(glowa);
        break;

            case '4':
            cout << "Podaj indeks: ";
            cin >> indeks;
            wyszukiwanie(glowa, indeks);
            break;

        case '5': reset(glowa);
        cout<< "Zresetowano liste studentow." << endl;
        drukowanie(glowa);
        break;

            case '6':
            cout << "Podaj indeks: ";
            cin >> indeks;
            if (!usuwanie_zadanego(&glowa, indeks))
            cout << "Nie odnaleziono studenta z podanym indeksem" << endl;
            else
             drukowanie(glowa);
            cout << "Konto USOS z podanym indeksem zostalo usuniete" << endl;
            break;

        case '7': zliczanie(glowa);
        break;

            case 'X':
            return 0;
            break;

        default:
        cout << "Komunikat nie jest jasny. Sprobuj ponownie."<<endl;
        break;


    }
}
return 0;
}
```