Sprawozdanie

Zadanie projektowe numer 3

Kacperek Chmurcia
Inżynieria i analiza danych, grupa P1

1 OPIS PROBLEMU

Dokonaj implementacji struktury danych typu lista dwukierunkowa wraz z wszelkimi potrzebnymi operacjami charakterystycznimi dla tej struktury (inicjowanie struktury, dodawanie/usuwanie elementow, wyswietlanie elementow, zliczanie elementow/wyszukiwanie zadanego elementu itp.)

2 LISTY

2.1 LISTA DWUKIERUNKOWA

Rysunek 1 Lista dwukierunkowa

Lista dwukierunkowa – w każdym elemencie listy jest przechowywane odniesienie zarówno do następnika, jak i poprzednika elementu w liście. Taka reprezentacja umożliwia swobodne przemieszczanie się po liście w obie strony. Zaletą takiego rozwiazania jest fakt, iż mozemy poruszac sie zarowno do przodu jak i do tylu po naszje liscie co znacząco oszczedza czas. Wadą chociażby w stosunku do listy jednokierunkowej jest to ze lista dwukierunkowa zajmuje wiecej pamieci poniewaz musi przechowac informacje o swoim poprzedniku jak i nastepcy.

2.2 LISTA JEDNOKIERUNKOWA

Lista jednokierunkowa

Lista jednokierunkowa jest strukturą o dynamicznie zmieniającej się wielkości. Listę można opisać jako uszeregowany zbiór elementów. Każdy element zawiera jakieś dane oraz wskazuje na swojego następcę. Cechą listy jednokierunkowej jest to, że można przeglądać ją tylko w jedną stronę, od początku do końca.

2.3 LISTA DWU/JEDNOKIERUNKOWA CYKLICZNA

Listy te odrózniaja się od poprzedniczek tym ze mozemy je przechodzic cyklicznie. Dokonuje sie tego ustawiajac pierwszy element listy jako nastepce ostatniego oraz ustawiajac ostatni element listy jako poprzednika pierwszego.

Lista dwukierunkowa cykliczna

Lista jednokierunowa cykliczna

2.4 WSKAZNIKI I LICZNIK

Tworząc listę w pamięci zwykle dodatkowo rezerwuje się trzy zmienne dla jej obsługi:

-wskaźnik head – wskazuje pierwszy element listy (ang. head = głowa)

-wskaźnik tail – wskazuje ostatni element listy (ang. tail = ogon)

-licznik count – zlicza elementy na liście

2.5 LISTA A TABLICA

2.5.1 Porównanie

Tablica jest niewątpliwe alternatywa dla listy. Tak samo jak w liscie mozemy bez problemu dodac element na koniec tablicy, problem pojawia sie gdy chcemy dodac element, w środku bądź na początek tablicy wtedy konieczne jest przesunięcie wszystykich elementów, które mają występowac po naszym dodawanym elemencie. ten pojawia się rownież, gdy chcemy usunąć jakiś element wtedy też musimy przesuwać elementy, aby zapełnić luke powstałą w wyniku usuwania.

2.5.2 Wady i zalety

Zalety:

- prosta nawigacja wewnątrz tablicy

- szybki dostęp do konkretnego elementu o konkretynm numerze

- większa odporność na błędy

Wady:

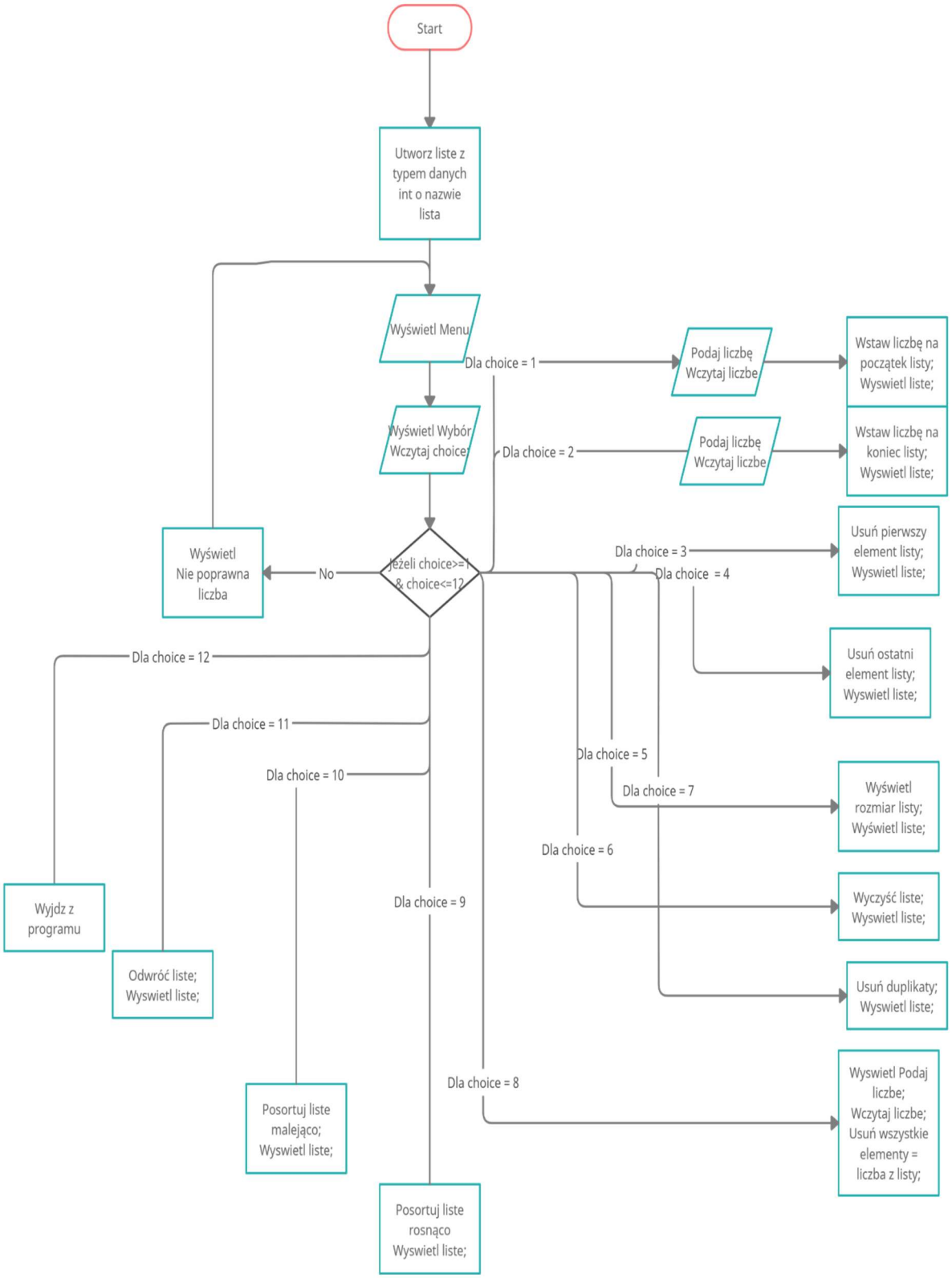
- niska elastyczność

- liniowa złożoność jeśli chodzi o operacje wstawiania i usuwania co za tym

Podsumywując tablica jest dobrą altenrnatywą dla listy jeśli bedzie ona wykorzystywana do przechowywania i odczytywania danych, na których nie będziemy wykonywać operacji. Jest szybsza w odczycie oraz znacznie stabilniejsza, lecz brak jej elastyczności jest znaczącą wadą.

3 PROGRAM

3.1 SCHEMAT BLOKOWY



Schemat blokowy

3.2 PSEUDOKOD

Utwórz liste z typem danych int o nazwie lista

Wyświetl Menu

Wyświetl „Wybor”

Wczytaj choice

Jeżeli choice = 1 to:

    Wyświetl „Podaj liczbe”

    Wczytaj liczbe

    Wstaw liczbe na początek listy

    Wyświetl liste

Jeżeli choice = 2 to:

    Wyświetl „Podaj liczbe”

    Wczytaj liczbe

    Wstaw liczbe na koniec listy

    Wyświetl liste

Jeżeli choice = 3 to:

    Usuń pierwszy element listy

    Wyświetl liste

Jeżeli choice = 4 to:

    Usuń ostatni element listy

    Wyświetl liste

Jeżeli choice = 5 to:

    Wyświetl rozmiar listy

    Wyświetl liste

Jeżeli choice = 6 to:

Wyczyść liste

Wyświetl liste

Jeżeli choice = 7 to:

Usuń duplikaty

Wyświetl liste

Jeżeli choice = 8 to:

Wyświetl podaj liczbe

Wczytaj liczbe

Usuń wszystkie elementy z list równe liczbie

Jeżeli choice = 9 to:

Posortuje liste rosnąco

Wyświetl liste

Jeżeli choice = 10 to:

Posotruj liste malejaco

Wyświetl liste

Jeżeli choice = 11 to:

Odwróć liste

Wyświetl liste

Jeżeli choice = 12 to:

Wyjdź z programu

Jeżeli choice rózne od liczb z przedziału <1,12> to:

Wyświetl Podales nie poprawna liczbe poda liczby z zakresu 1-12!

3.3 WŁASNY PROGRAM I KILKA NAJWAŻNIEJSZYCH FUNKCJONALNOŚCI

3.3.1 Utworzenie list oraz funkcji wyświetlenie listy z zapisem do pliku

in.cpp

```
#include <iostream>
#include <list>
#include <windows.h>
#include<string>
#include<fstream>
using namespace std;

list <int> lista; //utworzenie listy z podstawowy typem danych int
int choice,ile,a;

void wyswietl()
{
    ofstream zapis("lista_dwukierunkowa.txt");
    cout<<endl;
    cout<<" Lista: "<<endl;
    cout<<"________________________________"<<endl;

    for(list<int>::iterator i=lista.begin(); i!= lista.end(); ++i)
    {
        cout<<*i<<" ";
        zapis<<*i<<" ";

    }
```

3.3.2 Dodawanie Elementów

```
void push_front()
{
    int liczba;
    cout<<"Podaj ile liczb chcesz dodac: ";
    cin>>ile;
    for(int i=0;i<ile;i++){
        cout<<"Podaj jaka liczbe wstawic na poczatek listy: ";
        cin>>liczba;
        lista.push_front(liczba);
    }

    wyswietl();
}

void push_back()
{
    int liczba;
    cout<<"Podaj ile liczb chcesz dodac: ";
    cin>>ile;
    for(int i=0;i<ile;i++)
        {
        cout<<"Podaj jaka liczbe wstawic na koniec listy: "; //
        cin>>liczba;
        lista.push_back(liczba);
        }
    wyswietl();
}
```

3.3.3 Usuwanie elementów

```
}
void pop_front()
{
    cout<<"Podaj ile liczb chcesz usunac z poczatku listy: ";
    cin>>ile;
    for(int i=0;i<ile;i++)
    {
      lista.pop_front();
    }

    cout<<"Usunieto pierwszy element/elementy listy";
    wyswietl();
}
void pop_back()
{
    cout<<"Podaj ile liczb chcesz usunac z konca listy: ";
    cin>>ile;
    for(int i=0;i<ile;i++)
    {
      lista.pop_back();
    }
    cout<<"Usunieto ostatni element/element listy";
    wyswietl();
}
```

3.3.4 Zliczanie elementów

```
void ilejest()
{
    int szukana, licznik=0;
    bool jest=false;
    cout << "Podaj liczbe: ";
    cin >> szukana;
    for( list<int>::iterator i=lista.begin(); i!= lista.end(); ++i )
       {
          if (*i == szukana)                                    //funkcja spraw
        {
            jest=true;
            licznik++;
        }

       }
       if(jest==false)
    {
        cout<<"Nie ma podanej liczby w tablicy"<<endl;
    }else
    {
    cout << "Liczba " << szukana << " wystapila: " << licznik << " razy." << endl;
    }

    wyswietl();
}
```

3.3.5 Wyszukiwanie zadanego elementu

```
void czyjest()
{
    int szukana, licznik=0;
    bool jest=false;
    cout << "Podaj liczbe: ";
    cin >> szukana;
    cout << "Pozycja: ";
    for( list<int>::iterator i=lista.begin(); i!= lista.end(); ++i )
    {
        licznik++;
        if(*i == szukana)
        {
         jest=true;
         cout << licznik<<" ";
        }
    }
    if(jest==false)
     {
         cout<<"Nie ma podanej liczby w tablicy"<<endl;
     }
    cout<<endl<<endl;
     wyswietl();
}
```

3.3.6 Sortowanie malejące

```
void sortmal()
{
     cout<<"Posortowano liste malejaco";
     lista.sort();
     lista.reverse();                                    //funkc
     wyswietl();
}
```

3.3.7 Menu

```
while(choice!=14)
{
cout << "1.Dodaj element/elementy na poczatek listy"<<endl;
cout << "2.Dodaj element/elementy na koniec listy"<<endl;
cout << "3.Usun pierwszy element/element listy"<<endl;
cout << "4.Usun ostatni element/element listy"<<endl;
cout << "5.Pokaz rozmiar listy"<<endl;
cout << "6.Wyczysc liste"<<endl;
cout << "7.Usun duplikaty"<<endl;                              //MENU
cout << "8.Usun wszystkie elementy rowne: "<<endl;
cout << "9.Posortuj liste rosnaco"<<endl;
cout << "10.Posortuj liste malejaco"<<endl;
cout << "11.Odwroc liste"<<endl;
cout << "12.Znajdz pozycje zadanej liczby"<<endl;
cout << "13.Policz ile razy zadana liczba wystapila w liscie"<<endl;
cout << "14.Wyjdz"<<endl;
cout << "Wybor: ";
cin >> choice;
    switch (choice)
    {
        case 1:  push_front(); break;
        case 2:  push_back();  break;
        case 3:  pop_front();  break;
        case 4:  pop_back();   break;
        case 5:  size();       break;
        case 6:  clear();      break;                  //Wywolywanie funkcji
        case 7:  unique();     break;
        case 8:  remove();     break;
        case 9:  sort();       break;
        case 10: sortmal();    break;
        case 11: reverse();    break;
        case 12: czyjest();    break;
        case 13: ilejest();    break;
        case 14: exit();       break;
        default:
        cout<<"Podales nie poprawna liczbe podaj liczby z zakresu 1-12!";
        break;
    }
```

4 PODSUMOWANIE

Listy dwukierunkowe choć nieco bardziej skąplikowane od jednokierunkowy, co za tym idzie szybsze, maja wadę jaką jest zajmowanie wiekszej ilosci pamieci ponieważ potrzebny jest jeszcze im dodatkowy rekord na „poprzednika-prev”. Implementując liste programista powinien odpowiedziec sobie na pytanie czy potrzebuje prędkości w odczycie danych czy woli postawic na wolniejsza liste, lecz z mniejszym zużyciem pamieci.

5 SPIS TREŚCI