

Imię i Nazwisko	Kierunek	Rok studiów i grupa
Patryk Kozłowski	Informatyka Techniczna	1 rok grupa II
Data Zajęć	Numer i temat sprawozdania	
16.10.2024	5. Grafy	

Wprowadzenie teoretyczne

Grafy są strukturami danych składającymi się z wierzchołków i krawędzi, które łączą te wierzchołki. Umożliwiają modelowanie różnych typów sieci, jak również zależności między elementami systemu. Grafy mogą być nieskierowane, gdzie krawędzie są symetryczne, lub skierowane, z krawędziami działającymi w jednym kierunku. Dodatkowo, grafy mogą posiadać wagi, które przypisują określoną wartość do krawędzi, umożliwiając modelowanie odległości lub kosztów przejść między wierzchołkami. Struktura grafów znajduje szerokie zastosowanie w analizie sieci społecznych, mapach geograficznych, i innych systemach o skomplikowanych połączeniach.

Aby przechowywać i przetwarzać grafy, używamy różnych metod reprezentacji, takich jak macierz sąsiedztwa, macierz incydencji oraz listy sąsiedztwa. Macierz sąsiedztwa jest kwadratową macierzą, w której każdy wiersz i kolumna odpowiadają wierzchołkom grafu, a wartości określają połączenia między nimi. Wskaźniki, z kolei, są specjalnymi zmiennymi przechowującymi adresy w pamięci. W kontekście dynamicznej alokacji pamięci, wskaźniki odgrywają kluczową rolę, umożliwiając elastyczne tworzenie struktur danych o zmiennym rozmiarze, takich jak tablice dynamiczne.

Operacje odczytu i zapisu danych do plików pozwalają na przechowywanie i późniejsze odtwarzanie grafów oraz ich macierzy. Użycie bibliotek takich jak w C++ umożliwia zapis danych do pliku i ich odczyt, co jest niezbędne do utrwalania danych grafowych. Przykłady kodu przedstawiają, jak można zapisywać oraz odczytywać wartości z plików tekstowych, a także, jak implementować funkcje operujące na plikach za pomocą `ios::out` dla zapisu oraz `ios::in` dla odczytu.

Zadanie 1

In []: `//01.cpp`

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

using namespace std;

int main(){
    int n, m;
    fstream zapis;

    zapis.open("tablica.txt", ios::out);

    cout << "podaj rozmiar wierszy: " << endl;
```

```

cin >> n;

cout << "podaj rozmiar kolumn: " << endl;
cin >> m;

int **tab;

tab = new int *[n];

for (int i = 0; i < n; i++){
    tab[i] = new int[m];
}

for (int i = 0; i < n; i++){
    for (int j = 0; j < m; j++){
        tab[i][j] = (i + 1) * (j + 1);
    }
}

for (int i = 0; i < n; i++){
    for (int j = 0; j < m; j++){
        zapis << tab[i][j] << " ";
    }
    zapis << endl;
}

for (int i = 0; i < n; i++){
    delete []tab[i];
}

delete []tab;

zapis.close();
return 0;
}

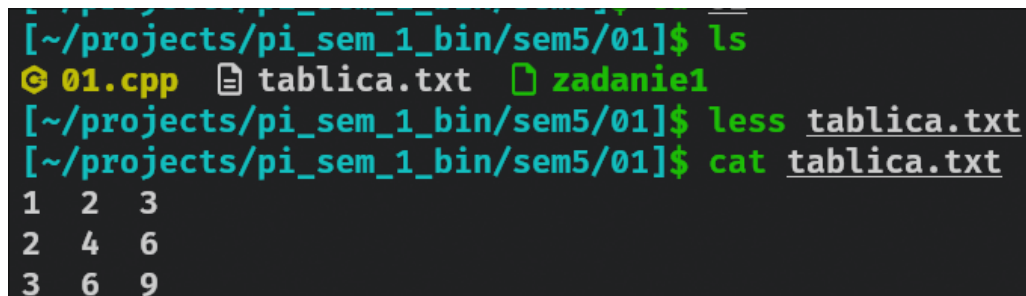
```

In []: // tablica.txt

```

1 2 3
2 4 6
3 6 9

```



```

[~/projects/pi_sem_1_bin/sem5/01]$ ls
01.cpp  tablica.txt  zadanie1
[~/projects/pi_sem_1_bin/sem5/01]$ less tablica.txt
[~/projects/pi_sem_1_bin/sem5/01]$ cat tablica.txt
1 2 3
2 4 6
3 6 9

```

Zadanie 2

In []: //02.cpp

```
#include <iostream>
```

```
#include <fstream>

using namespace std;

int main(){
    fstream odczyt;
    string h;
    string hs;

    odczyt.open("haslo.txt", ios::in);

    if (!odczyt.good()){
        cout << "nie ma pliku z haslami" << endl;
        return 1;
    }

    cout << "podaj haslo: ";
    cin >> h;

    odczyt >> hs;

    if (h == hs){
        fstream dziennik;
        dziennik.open("dziennik_ocen.txt", ios::in);
        string linia;

        while (getline(dziennik, linia)){
            cout << linia << endl;
        }
        dziennik.close();
    }

    odczyt.close();
    return 0;
}
```

In []: // dziennik_ocen.txt

```
polski 1 1 3 5
matematyka 1 1 5 5 4
fizyka 2 3 1 2
chemia 3 4 5
wf 1 1 1 1 1
```

In []: // haslo.txt

```
asd
```

```
[~/projects/pi_sem_1_bin/sem5/02]$ ls
02.cpp  dziennik_ocen.txt  haslo.txt  zadanie2
[~/projects/pi_sem_1_bin/sem5/02]$ ./zadanie2
podaj haslo: asd
polski 1 1 3 5
matematyka 1 1 5 5 4
fizyka 2 3 1 2
chemia 3 4 5
wf 1 1 1 1 1
[~/projects/pi_sem_1_bin/sem5/02]$ cat haslo.txt
asd%
[~/projects/pi_sem_1_bin/sem5/02]$ cat dziennik_ocen.txt
polski 1 1 3 5
matematyka 1 1 5 5 4
fizyka 2 3 1 2
chemia 3 4 5
wf 1 1 1 1 1 %
```

Zadanie 3

```
In [ ]: // 03.cpp

#include <iostream>

using namespace std;

int main(){
    int v, e;

    cout << "podaj liczbe wierzchołkow: ";
    cin >> v;

    int **sasiedzi = new int *[v];

    for (int i = 0; i < v; i++){
        sasiedzi[i] = new int[v];
    }

    for (int i = 0; i < v; i++){
        for (int j = 0; j < v; j++){
            sasiedzi[i][j] = 0;
        }
    }

    cout << "podaj liczbe krawedzi: ";
    cin >> e;

    for (int i = 0; i < e; i++){
        int a, b;
        cout << "podaj " << i + 1 << " pare krawedzi np. 1 3 :";
        cin >> a >> b;
        sasiedzi[a][b] = 1;
    }

    for (int i = 0; i < v; i++){
```

```

        for (int j = 0; j < v; j++){
            cout << sasiedzi[i][j] << " ";
        }
        cout << endl;
    }

    for (int i = 0; i < v; i++){
        delete []sasiedzi[i];
    }
    delete []sasiedzi;
    return 0;
}

```

```

[~/projects/pi_sem_1_bin/sem5/03]$ ls
03.cpp  zadanie3
[~/projects/pi_sem_1_bin/sem5/03]$ ./zadanie3
podaj liczbe wierzchołkow: 2
podaj liczbe krawedzi: 2
podaj 1 pare krawedzi np. 1 3 :1 1
podaj 2 pare krawedzi np. 1 3 :0 1
0 1
0 1
[~/projects/pi_sem_1_bin/sem5/03]$ |

```

Zadanie 4

```

In [ ]: // 04.cpp

#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main(){
    fstream odczyt;
    odczyt.open("dane.txt", ios::in);
    string pierwsze_dane;

    getline(odczyt, pierwsze_dane);

    string v_raw, e_raw;

    bool space_happened = false;

    for (int i = 0; i < pierwsze_dane.length(); i++){
        if (pierwsze_dane[i] == ' '){
            space_happened = true;
        }

        if (!space_happened && pierwsze_dane[i] != ' '){
            v_raw += pierwsze_dane[i];
        }
    }
}

```

```
    }
    else{
        e_raw += pierwsze_dane[i];
    }
}
int v_count, e_count;

v_count = stoi(v_raw);
e_count = stoi(e_raw);

int **sasiedzi = new int *[v_count];

for (int i = 0; i < v_count; i++){
    sasiedzi[i] = new int[v_count];
}

for (int i = 0; i < v_count; i++){
    for (int j = 0; j < v_count; j++){
        sasiedzi[i][j] = 0;
    }
}

string line;
string e1,e2;
int ie1, ie2;
space_happened = false;
while (getline(odczyt, line)){
    e1 = "";
    e2 = "";
    space_happened = false;
    for (int i = 0; i < line.length(); i++){
        if (line[i] == ' '){
            space_happened = true;
        }

        if (!space_happened && line[i] != ' '){
            e1 += line[i];
        }
        else if (space_happened && line[i] != ' '){
            e2 += line[i];
        }
    }
    ie1 = stoi(e1);
    ie2 = stoi(e2);
    sasiedzi[ie1][ie2] = 1;
}

for (int i = 0; i < v_count; i++){
    for (int j = 0; j < v_count; j++){
        cout << sasiedzi[i][j] << " ";
    }
    cout << endl;
}

for (int i = 0; i < v_count; i++){
    delete []sasiedzi[i];
}
delete []sasiedzi;
return 0;
}
```

```
In [ ]: // dane.txt
```

```
2 2  
0 1  
1 0  
0 0
```

```
[~/projects/pi_sem_1_bin/sem5/04]$ ls  
04.cpp dane.txt zadanie4  
[~/projects/pi_sem_1_bin/sem5/04]$ cat dane.txt  
2 3  
0 1  
1 0  
0 0%  
[~/projects/pi_sem_1_bin/sem5/04]$ ./zadanie4  
1 1  
1 0
```

Podsumowanie i Wnioski

Repozytorium github link: https://github.com/UmarlyPoeta/pi_sem_1_bin

Implementacja grafów i ich operacji w C++ jest realizowana poprzez różnorodne struktury danych i metody reprezentacji, takie jak macierze sąsiedztwa i wskaźniki.

Dynamiczna alokacja pamięci i operacje na plikach stanowią fundament w przetwarzaniu danych grafowych, zapewniając ich elastyczność oraz trwałość. Umiejętność manipulacji wskaźnikami i plikami umożliwia efektywne i zoptymalizowane przetwarzanie grafów, co jest kluczowe dla zaawansowanych zastosowań w algorytmice i analizie danych.