Jan Danowski

Politechnika Rzeszowska

Algorytmy i struktury danych

Sprawozdanie

Projekt 1

Opis problemu

Dla zadanej tablicy o rozmiarach MxN, pośród wszystkich jej podtablic o rozmiarach PxQ (P<M, Q<N), znajdź tę, dla której suma jej elementów jest maksymalna.

Przykład:

Wejście: [0 2 3 4 5],[1 3 4 5 3],[3 4 5 6 0]

P = 2, Q = 2

Wyjście:

[4 5]

[5 6]

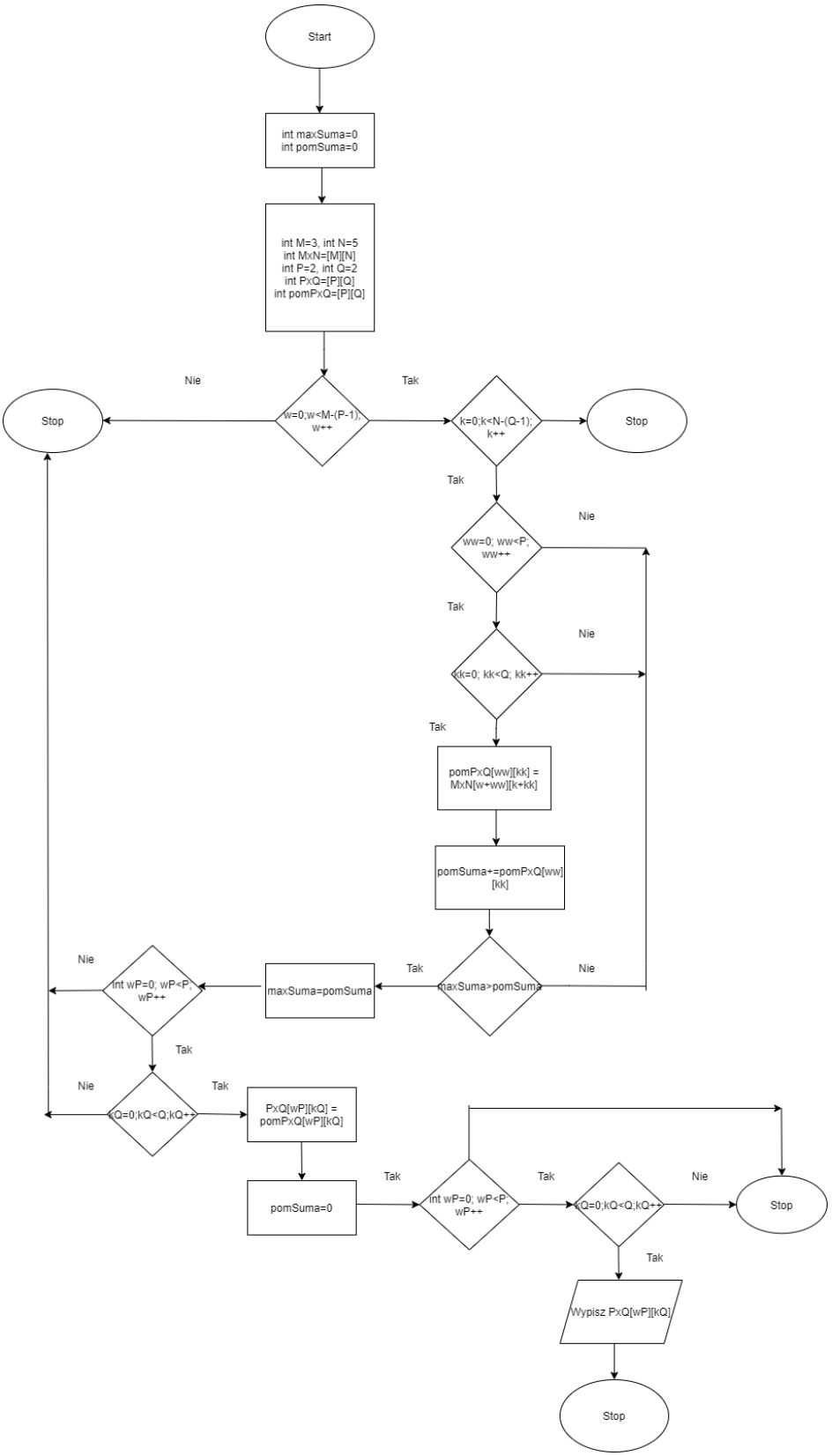
Opis podstaw teoretycznych

Tablice wielowymiarowe definiuje się za pomocą zapisu – typ_danych nazwa_tablicy [ ] [ ]; Elementy tablicy wielowymiarowej umieszczone są kolejno w pamięci komputera tak, że najszybciej zmienia się najbardziej skrajny prawy indeks. Oznacza to że tablica przechowywana jest rzędami.

Ważne jest to, aby w funkcjach mających jako parametry tablice wielowymiarową, zawsze podawać drugi wymiar tablicy. Do tablic wielowymiarowych w C++ przydatna jest też znajomość macierzy.

Tablice wielowymiarowe można również definiować za pomocą wektorów. Warto pamiętać, że tablice wielowymiarowe wywołujemy za pomocą pętli.

Schemat blokowy



Pseudokod

wczytaj zmienną maxSuma=0, pomSuma=0;

wczytaj M=3, N=5;

wczytaj MxN[3][5] = {0,2,3,4,5},{1,3,4,5,3},{3,4,5,6,0};

wczytaj P=2, Q=2;

wczytaj PxQ[2][2] = {0,0},{0,0};

Dla (int ww=0; ww < P; ww++) i (int kk=0; kk < Q; kk++) **wykonuj**

pomPxQ[ww][kk] = MxN[w+ww][k+kk];

pomSuma += pomPxQ[ww][kk];

jeżeli (pomSuma > maxSuma)

maxSuma = pomSuma;

Dla (int wP=0; wP < P; wP++) i (int kQ=0; kQ < Q; kQ++) **wykonuj**

PxQ[wP][kQ] = pomPxQ[wP][kQ];

pomSuma=0;

Dla (int wP=0; wP < P; wP++) i (int kQ=0; kQ < Q; kQ++)

wypisz PxQ[wP][kQ]

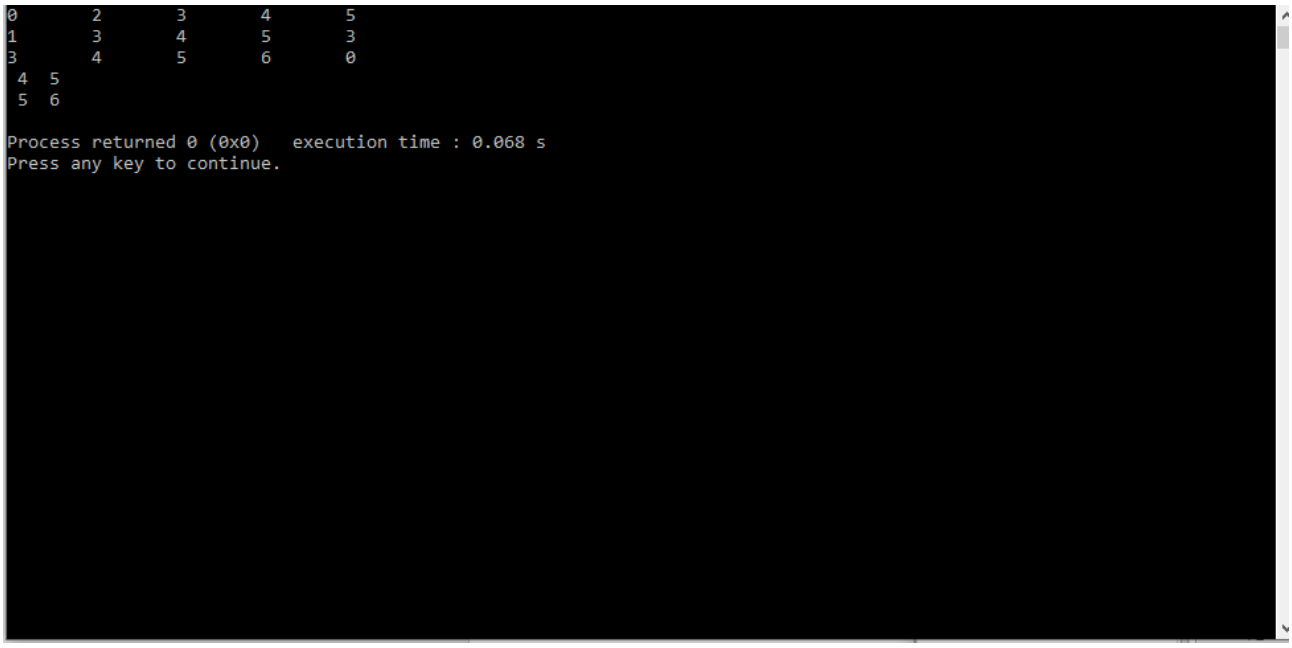
return 0;

Opis implementacji problemu

Definiujemy tablice główną oraz jej podtablice. Musimy zdefiniować zmienne pomocnicze oraz pętle pozwalające przechodzić po wierszach i kolumnach by móc później przepisać do tablicy PQ kolejne podtablice i liczyć ich sumę po kolei. Po wykonaniu pętli nastąpi przepisanie z tablicy MxN do pomocniczej zdefiniowanej jako pomPxQ oraz nastąpi zwiększanie jej sumy. Jeżeli nasza suma pomocnicza będzie większa od zmiennej maxSuma to dokonujemy podmiany wartości w tabeli PxQ. Na koniec wyświetlamy naszą znalezioną tablice pętlą for, gdyż jest to tablica wielowymiarowa.

Wnioski i podsumowanie

Podany algorytm na przykładzie z zadanie wykonuje się bardzo szybko, jest to czas rzędu 0.068s. Załączony jest tylko 1 sposób rozwiązania. Ze względu na napotkane problemy jest również brak wykresu złożoności czasowej, ale przeprowadzając jeden test na statycznej tablicy o rozmiarach 100x100, jestem w stanie stwierdzić, że czas obliczeń nie wzrasta znacząco. Jest to wzrost z 0.068s czyli naszego pierwszego czasu do 0.093s.



Rys. 1 Wynik programu

Bibliografia

https://e.kul.pl/files/10382/public/paip_w2_tabldwuwymiarowefcjerekurencyjne_1.pdf

Kod źródłowy

```
#include <iostream>
using namespace std;
void program(int maxSuma = 0, int pomSuma = 0)
{
   int M = 3, N = 5; //definiujemy tablicę bazową
   int MxN[3][5] =
  {{0,2,3,4,5},{1,3,4,5,3},{3,4,5,6,0}};
   int P = 2, Q = 2; //definiujemy tablicę PQ
   int  PxQ[2][2] = {{0,0},{0,0}};
   int  pomPxQ[2][2] = { {0,0},{0,0} };
   for(int w=0; w<3; w++) //ten for jest do wypisywanania tablicy
{
   for(int k=0; k<5; k++)
       {
       cout<<MxN[w][k]<<"\t";
       }
cout<<endl;
}
for (int w = 0; w < M-(P-1); w++)
{
for (int k = 0; k < N-(Q-1); k++)
{
for(int ww = 0; ww < P; ww++)

for (int kk = 0; kk < Q; kk++)
```

```
{
pomPxQ[ww][kk] = MxN[w+ww][k+kk];
pomSuma += pomPxQ[ww][kk];
}
if (pomSuma > maxSuma)
{
maxSuma = pomSuma;
for (int wP = 0; wP < P; wP++)
for (int kQ = 0; kQ < Q; kQ++)
{
PxQ[wP][kQ] = pomPxQ[wP][kQ];
}
}
pomSuma = 0;
}
}
for (int wP = 0; wP < P; wP++)
{
for (int kQ = 0; kQ < Q; kQ++)
{
cout <<" "<< PxQ[wP][kQ] << " ";
}
cout << endl;
}
}
int main()
{
program();

   return 0;
}
```