AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

ZAAWANSOWANE PROGRAMOWANIE

Macierz i Testowanie usługi Github Copilot

Autor:

Mateusz Stanek
Dawid Szołdra

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2024

Spis treści

1. Ogólne określenie wymagań

Celem projektu jest utworzenie programu implementującego klasę Macierzy w języku `C++`. Jedak, głównym celem projektu jest zapoznanie się z narzędziem GitHub Copilot i wyrobienie sobie o nim opinii, dlatego program powinien być przez niego w dużej części napisany.

Przewidywany jest działający program (w miarę możliwości modelu) i dokumentacja w Doxygenie.

2. Analiza problemu

2.1. Macierz

Macierz[1] jest dwu wymiarową tablicą elementów, reprezentującą odpowiednio ułożony zbiór wartości. Macierze często używane są w matematyce, fizyce i oczywiście informatyce, ze względu na fakt, że można na nich wykonywać różne operacje matematyczne.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Rys. 2.1. Przykładowa macierz

Na rys. 2.1 jest pokazana przykładowa macierz.

2.2. Git

Kolejnym konceptem, którym zajmuje się projekt jest narzędzie git[2]. Pozwala ono zarządzać poszczególnymi wersjami projektów. Głównym korzeniem gita jest system commitów, czyli zapisania zmian w pliku w stosunku do commita starszego. To, w połączeniu z jego innymi możliwościami pozwala na tworzenie długich i skomplikowanych osi czasu danych projektów.

Użycie gita można zademonstrować na prostym przykładzie. Tworzymy katalog a w nim repozytorium, uzywając komendy `git init`, jak widać na rys. 2.2.

```
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/mattys/skrypty-i-syfy/studia/rok2/progr
amowanie-zaawansowane/p1/git-test/.git/
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % ls
drwxr-xr-x - mattys  6 Nov 15:54 .git/
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test %
```

Rys. 2.2. Puste repozytorium git

Stwórzmy jakiś plik i dodajmy go do repozytorium. Plik można dodać do repozytorium komendą `git add`

```
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % echo "commit1" > plik.txt
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % git add plik.txt
```

Rys. 2.3. Stworznie pliku w repozytorium

Następnie należy scommitować zmiany.

```
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % git commit -am "utworzenie pl
ik.txt"
[master (root-commit) bb3b898] utworzenie plik.txt
 1 file changed, 1 insertion(+)
 create mode 100644 plik.txt
```

Rys. 2.4. Commit nr. 1

Na rysunku 2.4 użyta komenda `git commit` commituje wszystkie dodane pliki (`-a`) z jakimś komunikatem (`-m`).

```
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % echo "commit2" >> plik.txt
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % git commit -am "uzupelnienie
plik.txt"
[master 40694c2] uzupelnienie plik.txt
 1 file changed, 1 insertion(+)
```

Rys. 2.5. Commit nr. 2

Na rys. 2.5, został utworzony kolejny commit, dodajacy zmiany do `plik.txt`.

```
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % git log
commit 40694c2e73758368445cb817f4524ee5c5afbece (HEAD → master)
Author: mattys1 <mattys0082@gmail.com>
Date:   Wed Nov 6 16:26:07 2024 +0100

    uzupelnienie plik.txt

commit bb3b898df760395b5763575e204c3c43b513d2f6
Author: mattys1 <mattys0082@gmail.com>
Date:   Wed Nov 6 16:12:31 2024 +0100

    utworzenie plik.txt
```

Rys. 2.6. Log gita

Jak na rys. 2.6 jest pokazane, używając komendy `git log`, można wyświetlić log commitów w repozytorium.

```
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % git checkout bb3b898df760395b
5763575e204c3c43b513d2f6
Note: switching to 'bb3b898df760395b5763575e204c3c43b513d2f6'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at bb3b898 utworzenie plik.txt
~/skrypty-i-syfy/studia/rok2/programowanie-zaawansowane/p1/git-test % echo plik.txt
plik.txt
```

Rys. 2.7. Demonstracja checkout

Jak widać na rys. 2.7, komenda `git checkout`, pozwala na przejście repozytorium w inny stan, w tym przypadku przechodzi się do commita o danym ID, pokazanym na rys. 2.6. Jako, że jest to pierwszy commit, nie ma w nim zmian z drugiego.

2.3. Doxygen

Doxygen[3] jest narzędziem automatycznie generującym dokumentację programu z komentarzy w kodzie źródłowym. Potrafi on generować strony HTML, gdzie można dynamicznie nawigować się miedzy rożnymi częściami kodu oraz pliki LaTeX, które można konwertować na różne, statyczne formaty.

2.4. Github Copilot

GitHub Copilot[4] to model `LLM` oferowany przez GitHub - może on analizować kod źródłowy i funkcjonować jako zaawansowany autocompleter lub asystent potrafiący tworzyć proste fragmenty. Jest on bezpośrednio zintegrowany z wieloma narzędziami Microsoftu, takimi jak Visual Studio Code czy zwykłe Visual Studio. Jednak, jest on dostępny również jako rozszerzenie do innych edytorów, jak Neovim, którego instalacja przy użyciu menagera pluginów `Lazygit` jest ukazana na rys. 2.8.

```
return {
    "github/copilot.vim",
    config = function()
        vim.keymap.set('n', '<leader>cp', function ()
            vim.cmd('Copilot')
        end, { noremap = true, silent = true })
    end
}
```

Rys. 2.8. Plik instalacyjny Plugina Copilot

W Visual Studio jest on zainstalowany domyślnie.

3. Projektowanie

3.1. Implementacja macierzy

Do zaimplementowania Macierzy zostanie użyty Język `C++` z kompilatorem `g++` oraz `MSVC`. Wersja standardu `C++` to `C++23`. Jako, że projekt ma być rozdzielony na dwa pliki, zostanie zastosowany `CMake` w celu automatyzacji procesu budowania. `CMake` pozwala na generowanie plików budujących dany projekt, zgodnie z określoną konfiguracją. Oszczędza to programiście, szczególnie przy większych projektach, manualne pisanie Makefileów. Plik konfiguracyjny `CMakeLists.txt` może wyglądać jak na rysunku

```
1  cmake_minimum_required(VERSION 3.15)
2
3  set(PROJECT_NAME proj4)
4
5  project(${PROJECT_NAME} VERSION 0.1 LANGUAGES CXX)
6
7  set(CMAKE_CXX_STANDARD 23)
8  set(CMAKE_CXX_STANDARD_REQUIRED ON)
9  set(CMAKE_EXPORT_COMPILE_COMMANDS True)
10 set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_CURRENT_LIST_DIR}/out)
11
12 add_subdirectory(src)
13
```

Listing 1. Plik konfiguracyjny CMake

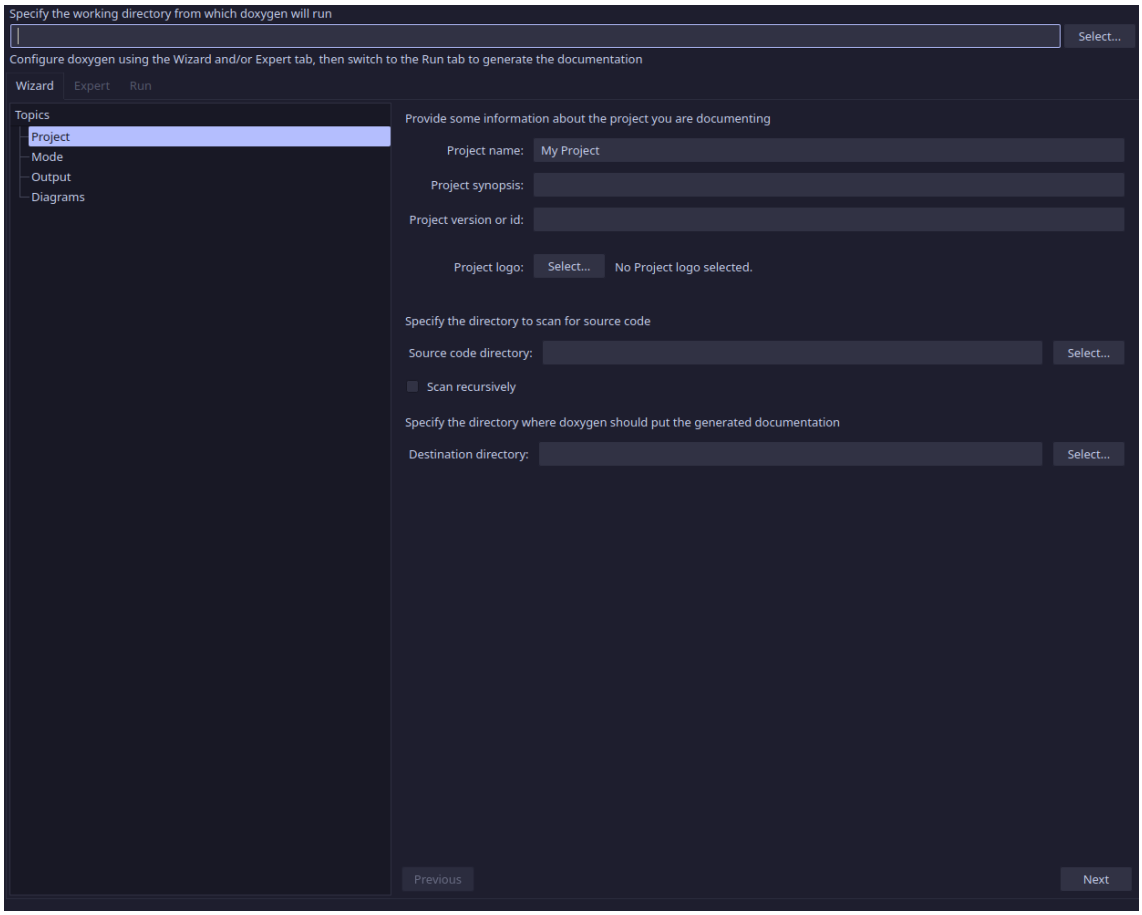
Edytorem będzie program Neovim oraz Visual Studio 2022 Community Edition. Jest to terminalowy edytor tekstu z możliwością poszerzenia funkcjonalności przy użyciu wszelkiego rodzaju pluginów. Wybrany został, dlatego że jest on już skonfigurowany na moim komputerze zgodnie z moimi preferencjami. Visual Studio jest zintegrowanym środowiskiem deweloperskim stworzonym i rozwijanym przez firmę Microsoft.

3.2. Git

Dla ułatwienia pracy, zastosowany został front-end dla gita o nazwie lazygit. Jest to terminalowy program, którego główną zaletą jest łatwa nawigacja przy użyciu klawiatury. Ponadto, jest on lekki i szybki.

3.3. Doxygen

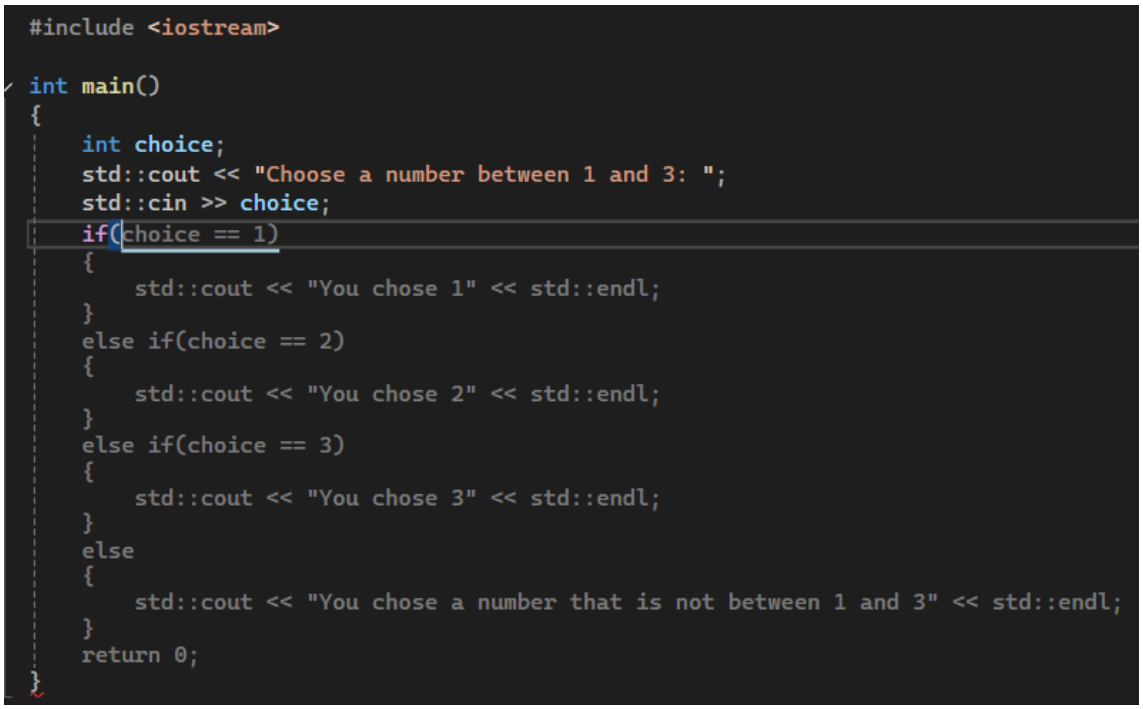
Konfiguracja dla Doxygena jest wygenerowana przy użyciu programu doxywizard, pokazany na rys. 3.1, pozwalającego na graficzne zmienianie ustawień. Po wygenerowaniu konfiguracji, Doxygen wywoływany jest przy użyciu komendy.



Rys. 3.1. Interfejs programu doxywizard

3.4. Github Copilot

Do napisania programu do pomocy został wykorzystany Github Copilot. Z pomocy kopilota można skorzystać na 3 sposoby. Sposób pierwszy - poprzez podpowiedzi generowane przez copilota które są oznaczone kolorem szarym, co pokazano na rysunku nr 3.2.

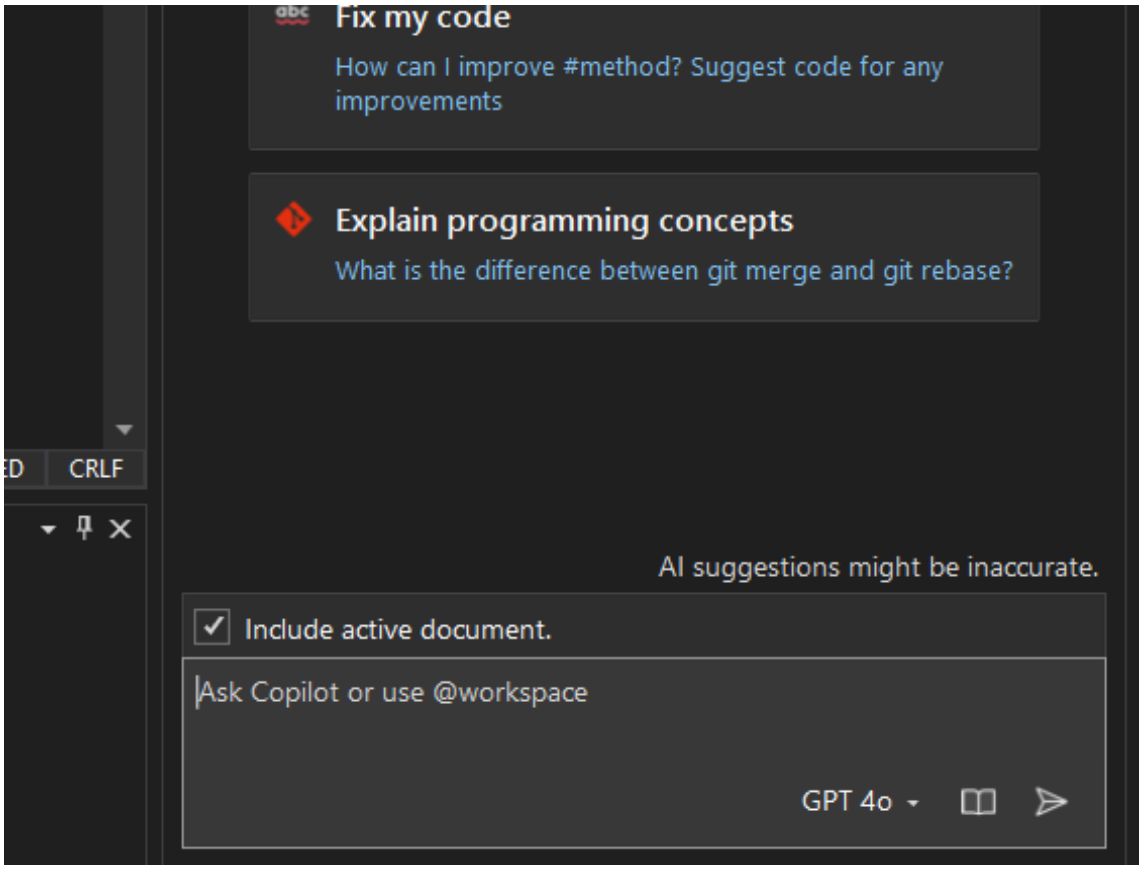


```cpp
#include <iostream>

int main()
{
    int choice;
    std::cout << "Choose a number between 1 and 3: ";
    std::cin >> choice;
    if(choice == 1)
    {
        std::cout << "You chose 1" << std::endl;
    }
    else if(choice == 2)
    {
        std::cout << "You chose 2" << std::endl;
    }
    else if(choice == 3)
    {
        std::cout << "You chose 3" << std::endl;
    }
    else
    {
        std::cout << "You chose a number that is not between 1 and 3" << std::endl;
    }
    return 0;
}
```

Rys. 3.2. Sugerowanie przez copilot

Sposób drugi to standardowe okienko chat gdzie można zapytać go o rzeczy związane z kodem, przedstawione zostało na rysunku nr 3.3



Rys. 3.3. okienko chat Copilot

Trzeci sposób to wykorzystanie komentarzy jako poleceń co ma zrobić copilot, jak pokazano na rysunku nr 3.4

```
// Funkcja kalkulatora dodająca, odejmująca, mnożąca i dzieląca dwie liczby
int kalkulator(int a, int b, char dzialanie)
{
    if (dzialanie == '+')
    {
        return a + b;
    }
    else if (dzialanie == '-')
    {
        return a - b;
    }
    else if (dzialanie == '*')
    {
        return a * b;
    }
    else if (dzialanie == '/')
    {
        return a / b;
    }
    else
    {
        return 0;
    }
}
```

Rys. 3.4. Zastosowanie komentarzy do generowania kodu

4. Implementacja

4.1. Klasa Matrix

Klasa jest zaimplementowana jako jeden plik `.hpp`. Nie jest podzielona na plik implementacji oraz nagłówek, ponieważ jest ona szablonem. Deklaracja klasy oraz prywatne elementy wyglądają tak jak na listingu nr. 2.

```
template <typename T>
class Matrix {
  private:
  std::vector<std::vector<T>> data;

  public:

  Matrix(void) {}

  Matrix(int n) : data(n, std::vector<T>(n)) {}

  Matrix(int n, int* t) : data(n, std::vector<T>(n)) {
    for(auto& row : data){
      row = {t, t + n};
      t += n;
    }
  }

  Matrix(Matrix const& m) : data(m.data) {}

  void wypisz(void) {
    for(const auto& row : data){
      for(const auto& elem : row) {
        std::print("{} ", elem);
      }
      std::println();
    }
  }

  Matrix& alokuj(int n){
    if(data.size() == 0){
      data.resize(n, std::vector<T>(n));
    } else {
      if(data.size() < n){
        data.resize(n, std::vector<T>(n));
      }
    }
```

```
38      return *this;
39    }
40
41    int pokaz(int x, int y) const {
42      return data[x][y];
43    }
44
45    Matrix& dowroc(void) {
46      Matrix<T> temp(data.size());
47
48      for(int i = 0; i < data.size(); i++){
49        for(int j = 0; j < data.size(); j++){
50          temp.data[i][j] = data[j][i];
51        }
52      }
53
54      *this = temp;
55      return *this;
56    }
57
58    Matrix& losuj(void) {
59      std::random_device rd;
60      std::mt19937 gen(rd());
61      std::uniform_int_distribution dis(0, 9);
62
63      for(int i = 0; i < data.size(); i++){
64        for(int j = 0; j < data.size(); j++){
65          data[i][j] = dis(gen);
66        }
67      }
68      return *this;
69    }
70
71    Matrix& losuj(int x) {
72      if (data.size() == 0) return *this;
73
74      std::random_device rd;
75      std::mt19937 gen(rd());
76      std::uniform_int_distribution<> dis(0, 9);
77
78      int n = (int)data.size();
79      for (int i = 0; i < x; i++) {
80        int a = dis(gen) % n;
81        int b = dis(gen) % n; // j.w.
82        data[a][b] = dis(gen);
```

```
83        }
84        return *this;
85      }
86
87      Matrix& diagonalna(const int* t) {
88        for(int i = 0; i < data.size(); i++){
89          for(int j = 0; j < data.size(); j++){
90            if(i == j){
91              data[i][j] = t[i];
92            } else {
93              data[i][j] = 0;
94            }
95          }
96        }
97        return *this;
98      }
99
100     Matrix& diagonalna_k(int k, const int* t) {
101       for(int i = 0; i < data.size(); i++){
102         for(int j = 0; j < data.size(); j++){
103           if(i == j + k){
104             data[i][j] = t[j];
105           } else {
106             data[i][j] = 0;
107           }
108         }
109       }
110       return *this;
111     }
112
113     Matrix& kolumna(int x, const int* t) {
114       for(int i = 0; i < data.size(); i++){
115         data[i][x] = t[i];
116       }
117       return *this;
118     }
119
120     Matrix& wiersz(int x, const int* t) {
121       for(int i = 0; i < data.size(); i++){
122         data[x][i] = t[i];
123       }
124       return *this;
125     }
126
127     Matrix& przekatna(void) {
```

```
128       for(int i = 0; i < data.size(); i++){
129         for(int j = 0; j < data.size(); j++){
130           if(i == j){
131             data[i][j] = 1;
132           } else {
133             data[i][j] = 0;
134           }
135         }
136       }
137       return *this;
138     }
139
140     Matrix& pod_przekatna(void) {
141       for(int i = 0; i < data.size(); i++){
142         for(int j = 0; j < data.size(); j++){
143           if(i > j){
144             data[i][j] = 1;
145           } else {
146             data[i][j] = 0;
147           }
148         }
149       }
150       return *this;
151     }
152
153     Matrix& nad_przekatna(void) {
154       for(int i = 0; i < data.size(); i++){
155         for(int j = 0; j < data.size(); j++){
156           if(i < j){
157             data[i][j] = 1;
158           } else {
159             data[i][j] = 0;
160           }
161         }
162       }
163       return *this;
164     }
165
166     Matrix& wstaw(int x, int y, int wartosc) {
167       data[x][y] = wartosc;
168       return *this;
169     }
170
171     Matrix& szachownica(void) {
172       for (int i = 0; i < (int)data.size(); i++) {
```

```
173         for (int j = 0; j < (int)data.size(); j++) {
174           data[i][j] = ((i + j) % 2 == 0) ? 0 : 1;
175         }
176       }
177       return *this;
178     }
179
180     Matrix& operator+(Matrix const& m) {
181       if (data.size() != m.data.size()) {
182         return *this;
183       }
184
185       for (int i = 0; i < (int)data.size(); i++) {
186         for (int j = 0; j < (int)data.size(); j++) {
187           data[i][j] += m.data[i][j];
188         }
189       }
190       return *this;
191     }
192
193     Matrix& operator*(Matrix const& m) {
194       int n = (int)data.size();
195       if (n == 0 || n != (int)m.data.size()) {
196         return *this;
197       }
198
199       Matrix<T> result(n);
200       for (int i = 0; i < n; i++) {
201         for (int j = 0; j < n; j++) {
202           T sum = 0;
203           for (int k = 0; k < n; k++) {
204             sum += data[i][k] * m.data[k][j];
205           }
206           result.data[i][j] = sum;
207         }
208       }
209       *this = result;
210       return *this;
211     }
212
213     Matrix& operator+(int a) {
214       for (auto& row : data) {
215         for (auto& elem : row) {
216           elem += a;
217         }
```

```
218       }
219       return *this;
220     }
221
222     Matrix& operator*(int a) {
223       for (auto& row : data) {
224         for (auto& elem : row) {
225           elem *= a;
226         }
227       }
228       return *this;
229     }
230
231     Matrix& operator-(int a) {
232       for (auto& row : data) {
233         for (auto& elem : row) {
234           elem -= a;
235         }
236       }
237       return *this;
238     }
239
240     friend Matrix operator+(int a, Matrix const& m) {
241       Matrix result(m);
242       for (auto& row : result.data) {
243         for (auto& elem : row) {
244           elem = a + elem;
245         }
246       }
247       return result;
248     }
249
250     friend Matrix operator*(int a, Matrix const& m) {
251       Matrix result(m);
252       for (auto& row : result.data) {
253         for (auto& elem : row) {
254           elem = a * elem;
255         }
256       }
257       return result;
258     }
259
260     friend Matrix operator-(int a, Matrix const& m) {
261       Matrix result(m);
262       for (auto& row : result.data) {
```

```
263         for (auto& elem : row) {
264           elem = a - elem;
265         }
266       }
267       return result;
268     }
269
270     Matrix& operator++(int) {
271       for (auto& row : data) {
272         for (auto& elem : row) {
273           elem += 1;
274         }
275       }
276       return *this;
277     }
278
279     Matrix& operator--(int) {
280       for (auto& row : data) {
281         for (auto& elem : row) {
282           elem -= 1;
283         }
284       }
285       return *this;
286     }
287
288     Matrix& operator+=(int a) {
289       for (auto& row : data) {
290         for (auto& elem : row) {
291           elem += a;
292         }
293       }
294       return *this;
295     }
296
297     Matrix& operator-=(int a) {
298       for (auto& row : data) {
299         for (auto& elem : row) {
300           elem -= a;
301         }
302       }
303       return *this;
304     }
305
306     Matrix& operator*=(int a) {
307       for (auto& row : data) {
```

```
308         for (auto& elem : row) {
309           elem *= a;
310         }
311       }
312       return *this;
313     }
314
315     Matrix& operator()(double a) {
316       int val = (int)std::floor(a);
317       for (auto& row : data) {
318         for (auto& elem : row) {
319           elem += val;
320         }
321       }
322       return *this;
323     }
324
325     friend std::ostream& operator<<(std::ostream& o, Matrix const&
    m) {
326       for (const auto& row : m.data) {
327         for (const auto& elem : row) {
328           o << elem << " ";
329         }
330         o << "\n";
331       }
332       return o;
333     }
334
335     bool operator==(const Matrix& m) const {
336       if (data.size() != m.data.size() || data[0].size() != m.data
    [0].size()) return false;
337       for (int i = 0; i < (int)data.size(); i++) {
338         for (int j = 0; j < (int)data[i].size(); j++) {
339           if (data[i][j] != m.data[i][j]) return false;
340         }
341       }
342       return true;
343     }
344
345     bool operator>(const Matrix& m) const {
346       if (data.size() != m.data.size() || data[0].size() != m.data
    [0].size()) return false;
347       for (int i = 0; i < (int)data.size(); i++) {
348         for (int j = 0; j < (int)data[i].size(); j++) {
349           if (!(data[i][j] > m.data[i][j])) return false;
```

```
350         }
351       }
352       return true;
353     }
354
355     bool operator<(const Matrix& m) const {
356       if (data.size() != m.data.size() || data[0].size() != m.data
    [0].size()) return false;
357       for (int i = 0; i < (int)data.size(); i++) {
358         for (int j = 0; j < (int)data[i].size(); j++) {
359           if (!(data[i][j] < m.data[i][j])) return false;
360         }
361       }
362       return true;
363     }
364   };
```

Listing 2. Klasa `Matrix`

Klasa posiada tyko jedno prywatne pole, którym jest `std::vector<std::vector<T>> data`. oraz publiczne metody takie jak:

- Konstruktor domyslny
- Konstruktor z rozmiarem macierzy
- 3 metody podstawowe, takie jak wypisz, alokuj, pokaz.
- Pozostałe metody

Metody zostaną opisane poniżej.

- `Konstruktor domyślny`
  Domyślny konstruktor, nie wykonuje żadnych działań poza utworzeniem obiektu.
- `Konstruktor z rozmiarem`
  Znajduje się w wierszu 10. Jest odpowiedzialny za utworzenie macierzy n*n wypełnionej wartościami domyślnymi T data() alokuje pamięć dla wektora o długości n, gdzie każdy element to wektor o długości n.
- `Konstruktor z tablicą`
  Znajduje się w wierszach od 12 do 17. Jest odpowiedzialny za tworzenie macierzy o n*n wypełnionej wartościami t. Instrukcja row kopiuje n elementów z tablicy t do bieżącego wiersza row. Instrukcja t+=n przesuwa wskaźnik do następnego zestawu elementów.

- `Konstruktor kopiujący`
  Znajduje się w wierszu 19. Jest odpowiedzialny za tworzenie kopii macierzy n poprzez skopiowanie jej danych.

- `wypisz()`
  Znajduje się w wierszach od 21 do 28. Jest odpowiedzialna za wyświetlanie elementów macierzy w konsoli. Pętla zewnętrzna jest odpowiedzialna za przechodzenie przez wiersze macierzy a wewnętrzna przez elementy wiersza.

- `alokuj()`
  Znajduje się w wierszach od 30 do 39. Jest odpowiedzialna za alokowanie pamięci dla macierzy o rozmiarze n*n. Jeżeli macierz jest pusta to tworzy nową macierz, jeżeli jest mniejsza od n to zmienia jej rozmiar na n*n.

- `pokaz()`
  Znajduje się w wierszach od 41 do 43. Zwraca wartość elementu na pozycji x,y w macierzy.

- `dowroc()`
  Znajduje się w wierszach od 45 do 56. Jest odpowiedzialna za zamianę wierszy z kolumnami. Tworzy tymczasową macierz `temp` aby wypełnić ją wartościami transponowanymi względem macierzy wejściowej. Następnie zastępuje bieżącą macierz nowymi wartościami.

- `losuj()`
  Znajduje się w wierszach od 58 do 69. Tworzy generator liczb pseudolosowych od 0 do 9 a następnie przy pomocy pętli for wypełnia całą macierz losowymi wartościami z zakresu.

- `losuj_x_elementów()`
  Znajduje się w wierszach od 71 do 85. Zasada działania jest podobna do poprzedniej metody losuj z zakresu ale tutaj losujemy x elementów.

- `diagonalna()`
  Znajduje się w wierszach od 87 do 98. Tworzy wzór szachownicy, gdzie pola mają wartość albo 0 albo 1, dzieje się to przy pomocy warunku (i+j) modulo 2 == 0.

- `diagonalna_k()`
  Znajduje się w wierszach od 100 do 111, Jest odpowiedzialna za ustawienie wartości przekątnej przesuniętej o k miejsc.

- `kolumna()`
  Znajduje się w wierszach od 113 do 118. Jest odpowiedzialna za wypełnienie kolumny k wartościami z tablicy t.

- `wiersz()`
  Znajduje się w wierszach od 120 do 125. Jest odpowiedzialna za wypełnienie wiersza x wartościami z tablicy t.

- `przekatna()`
  Znajduje się w wierszach od 127 do 138. Jest odpowiedzialna za wypełnienie głównej przekątnej 1 a resztę 0.

- `pod_przekatna()`
  Znajduje się w wierszach od 140 do 151. Jest odpowiedzialna za wypełnienie obszaru pod główną przekątną 1 a resztę 0.

- `nad_przekatna()`
  Znajduje się w wierszach od 153 do 164. Jest odpowiedzialna za wypełnienie obszaru nad główną przekątną 1 a resztę 0.

- `wstaw()`
  Znajduje się w wierszach od 166 do 169. Jest odpowiedzialna za wstawienie wartości do macierzy na pozycji (x,y).

- `Szachownica()`
  Znajduje się w wierszach od 171 do 178.Jest odpowiedzialna za utworzenie wzoru szachownicy w macierzy.

- `Metody firend operator+(Matirx a)`
  Znajduje się w wierszach od 180 do 191. Jest odpowiedzialna za dodawanie macierzy do macierzy element po elemencie.

- `operator*(Matrix a)`
  Znajduje się w wierszach od 193 do 211. Jest odpowiedzialna za mnożenie macierzy przez inną macierz.

- `operator+(int a)`
  Znajduje się w wierszach od 213 do 220. Jest odpowiedzialna za dodawanie do każdego elementu macierzy liczby całkowitej.

- `operaotr*(int a)`
  Znajduje się w wierszach od 222 do 229. Jest odpowiedzialna za mnożenie każdego elementu macierzy przez liczbę całkowitą.

- `operator-(int a)`
  Znajduje się w wierszach od 231 do 238. Jest odpowiedzialna za odjęcie od każdego elementu macierzy liczby całkowitej.

- `operator+(int a, const n)`
  Znajduje się w wierszach od 240 do 248. Jest odpowiedzialna za dodawanie liczb całkowitej do każdego elementu macierzy(postać a+n).

- `operator*(it a, const n)`
  Znajduje się w wierszach od 250 do 258. Jest odpowiedzialna za mnożenie każdego elementu macierzy przez liczbę całkowitą(postać a*n).

- `operator-(int a, const n )` Znajduje się w wierszach od 260 do 268. Jest odpowiedzialna za odejmowanie każdego elementu macierzy od liczby a(postać a-n).

- `operator++`
  Znajduje się w wierszach od 270 do 277. Jest odpowiedzialny za zwiększenie każdego elementu o 1.

- `operator--`
  Znajduje się w wierszach od 279 do 284. Jest odpowiedzialna za zmniejszenie wartości elementu o 1.

- `operator +=(int a)`
  Znajduje się w wierszach od 288 do 295. Dodaje liczbę całkowitą do każdego elementu macierzy.

- `operator -=(int a)`
  Znajduje się w wierszach od 297 do 304. Jest odpowiedzialna za odejmowanie od każdego elementu macierzy liczby całkowitej.

- `operator()(double a)`
  Znajduje się w wierszach od 315 do 323. Jest odpowiedzialna za dodawanie do każdego elementu macierzy cząstki liczby całkowitej

5. Wnioski

Nie należy ignorować narzędzi AI jak Copilot i uznawać je jako jakieś tymczasowe zabawki dla deweloperów, które kiedyś wyjdą z mody. Nawet w swoim obecnym stanie, sam fakt, że po napisaniu `for`, jednym kliknięciem taba, *najprawdopodobniej* Copilot wytworzy nam sensowną pętle *biorąc pod uwagę kontekst szerszego programu* jest bez dwóch zdań bardzo użyteczny, po prostu przez to, że oszczędza to programiście czas. Ponadto, narzędzia te są świetnymi wyszukiwarkami, co szczególnie się tyczy modeli, które mają dostęp do internetu. Nie trzeba praktycznie walczyć z zareklamowanym po kark Google i przeszukiwać napisanych przez boty stron - taki model automatycznie "przesieje"internet i pokaże nam informacje, które faktycznie dotyczą naszego zapytania. No, chyba że o to co się pytamy jest rzeczą niszową.

Niektórym (najbardziej to inwestorom w firmy zajmujące się AI) wydaje się, że LLM może za człowieka myśleć. Po części to prawda - jeżeli przed modelem o danym problemie myśleli inni - im więcej głowili się tym lepiej - i te przemyślenia gdzieś upublicznili, to LLM błyskawicznie może sięgnąć po tego problemu rozwiązanie. Ba, może nawet je po części zmodyfikować. Problem pojawia się, jeżeli zaczniemy naciskać na biednego chatbota. Fakty są następujące: LLM potrzebuje bardzo dużej ilości informacji o danym koncepcie, aby go opanować oraz LLM słabo potrafi rozumować, to jest, syntezować znane już informacje w nowe. Praktycznie, objawia się to gdy zapytamy się czatbota o rzecz, o której się mało mówi.

Osobiście mogę przytoczyć przykład próby zrozumienia API `pipewire`[5]. Jest to projekt zajmujący się zarządzaniem strumieniami audio (takimi z aplikacji) na Linuxie. Rzecz, z natury niszowa. Chciałem przechwycić wyjście jednej aplikacji i uzyskać na bieżąco sample jakie wysyła. Ile to było bawienia się z ChatemGPT, prób wyplucia przez niego programu który się nie segfaultuje - głównie dlatego, że nie chciało mi się czytać dokumentacji. Czat podczas treningu pewnie przeczytał - ale mało co z tego wyciągnął, bo strona z API to było jedno z niewielu miejsc, gdzie owo API było opisane. W końcu okazało się, że na stronie był przykład programu, który robił prawie to samo co chciałem i jedyne co ChatGPT mi dał to powód do przeczytania dokumentacji.

Pytać się można czy LLMy zastąpią programistów. Wydaje mi się, że - jeżeli nie teraz to za parę lat - programiści, których praca składa się z kopiowania zapytań o najnowszy framework JS ze Sack Overflowa i wklejania, faktycznie są zagrożeni, bo LLMy właściwie robią to, ale szybciej i taniej. Lecz programiści, którzy faktycznie tworzą coś nowego, faktycznie tworzą projekty wcześniej niedokonane, stoją na czele innowacji lub nią są - nie mają się czym martwić na długo.

Co do samego Copilota, to używa się go przyjemnie jak sensownie pisze, choć nie lubię tego, że pokazuje tekst w tej samej linii co kod. Odwraca to uwagę podczas pisania, trzeba zatrzymać się, przeczytać co on sugeruje i najprawdopodobniej kontynuować z pisaniem tego co się chciało. Wolałbym gdyby podpowiedzi były w formacie zwykłego autocompletea jak np. snippety.

Bibliografia

[1] *Artykul Wikipedii o macierzy*. URL: https://en.wikipedia.org/wiki/Matrix_(mathematics).

[2] *Strona Gita*. URL: https://git-scm.com/.

[3] *Strona Doxygena*. URL: https://www.doxygen.nl/.

[4] *Strona GitHub Copilot*. URL: https://github.com/features/copilot.

[5] *Strona PipeWire*. URL: https://pipewire.org/.

Spis rysunków

Spis tabel

Spis listingów