WAŻNE FUNKCJE

Tablica 2-wymiarowa dynamiczna

> Stworzymy taką macierz:

```
1 2
3 4
5 6
```

```
int rows = 2; // Alternatywnie width
int colums = 3; // Alternatywnie height

pixels = new int*[colums];
for (int i=0; i<colums; i++)
    pixels[i] = new int[rows];

// Wypełnienie przykładowymi danymi
int value = 1;
for (int i = 0; i < colums; i++)
    for (int j = 0; j < rows; j++)
        pixels[i][j] = value++;

// Na samym końcu usuwanie z pamięci
for (int i=0; i<colums; i++)
    delete[] pixels[i];

delete[] pixels;
```

Obrazek

Przykładowa zawartość .pgm (co po // to komentarz):

```
P2 // Format P2 tekst jawny P5 kodowanie binarne
4 4 // szerokosc wysokosc
255 // maksymalna wartość
0  16  32  48 // cała macierz (odzwierciedlenie piksel po pikselu)
64  80  96  112
128 144 160 176
192 208 224 240
```

Kod odczytujący

```
#include <fstream>

void main() {
    string filepath = "sciezka_do_pliku.pgm"
    ifstream file(filepath);

    string format;
    int width, height, max_gray

    file >> format;
    file >> width >> height;
    file >> max_gray;

    pixels = new int*[height];
        for (int i=0; i<height; i++)
            pixels[i] = new int[width];

    for (int i = 0; i < image.height; ++i)
        for (int j = 0; j < image.width; ++j)
            file >> pixels[i][j];

    file.close();
}
```

Polskie znaki

```
if ((unsigned char)slowo[i] >= 192) polski znak → ma 2 bajty
else zwykły ASCII → 1 bajt
```

```
// Usuwa spacje i zamienia duze znaki na małe
string przygotowanie_slowa(string slowo) {
    string preparedWord = "";
    for (int i = 0; i < slowo.length(); i++) {
        unsigned char c = slowo[i];

        if (c == ' ')
            continue;

        // ASCII A-Z -> a-z
        if (c >= 'A' && c <= 'Z')
            c = c + 32;

        preparedWord += c;
    }
    return preparedWord;
```

```
}

bool porownanie(string slowo) {
    int i = 0;
    int j = slowo.length() - 1;

    while (i < j) {
        unsigned char a = slowo[i];
        unsigned char b = slowo[j];

        int lenA = (a < 128) ? 1 : 2; // ile bajtów zajmuje znak po lewej
        int lenB = (b < 128) ? 1 : 2; // ile bajtów zajmuje znak po prawej

        // ASCII zajmuje 1 bajt
        if (lenA == 1 && lenB == 1) { // 2 ASCII
            if (slowo[i] != slowo[j]) return false;
        }
        else if (lenA == 2 && lenB == 2) { // Polski znak zajmuje 2 bajty
            if (a != static_cast<unsigned char>(slowo[j-1]) ||
static_cast<unsigned char>(slowo[i+1]) != b )
                return false;
        }
        else { // ASCII i polski znak
            return false;
        }

        i += lenA;
        j -= lenB;
    }
    return true;
}
```

Funkcja przyjmuje stringa i sprawdza czy jest w nim chociaż jeden polski znak

```
bool czy_slowo_ma_polski_znak(string slowo) {
    for (int i=0; i<slowo.length(); i++) {
        int literka = static_cast<int>(slowo[i]);
        cout << "Literka " << slowo[i];
        if ((literka>=65 && literka<=90) || (literka>=97 && literka<=122))
            continue;
        else
            return true;
    }
```

```
    return false;
}
```

Funkcja sortuje pacjentów wg wagi

> Jeśli ma sortować malejąco >
> Jeśli ma sortować rosnąco <
>
> Jeśli ma sortować coś innego niż wage to zamień .waga (działają też stringi)

```
void sortowanie(Pacjent *pacjenci, int n) {
    for (int i=0; i<n; i++)
        for (int j=0; j<n-i-1; j++)
            if (pacjenci[j].waga > pacjenci[j+1].waga)
                swap(pacjenci[j], pacjenci[j+1]);
}
```

Funkcja rekurencyjna do wyszukiwania binarnego

> W przykładzie wyszukujemy nazwisko

```
bool wyszukiwanieRek(Pacjent *pacjenci, int lewy, int prawy, string nazwisko) {
    if (lewy > prawy)
        return false;

    int srodek = lewy + (prawy - lewy) / 2;

    if (pacjenci[srodek].nazwisko == nazwisko)
        return true;

    if (pacjenci[srodek].nazwisko < nazwisko) // szukanie w prawo
        return wyszukiwanieRekurencyjne(pacjenci, srodek + 1, prawy, nazwisko);
    else // szukanie w lewo
        return wyszukiwanieRekurencyjne(pacjenci, lewy, srodek - 1, nazwisko);
}
```