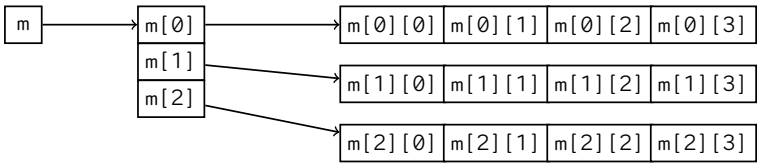
Objektno programiranje
Vježba 5 - dvodimenzionalna matrica

Da bi napravili Board teba nam dvodimenzionalna matrica. Nju možemo realizirati na nekoliko načina. Jedan od načina je deklarirati obično dvodimenzionalno polje `char matrix[rows][cols];`, te elementima pristupati običnom indeksnom notacijom `matrix[i][j];`. Međutim, memorija za takvo polje alocira se na **stacku**, a u zadatku piše da se memorija treba alocirati na **heapu**, dakle polje treba dinamički alocirati.

1. **Dinamička alokacija pomoću niza pointera.** C++ ne podržava višedimenzionalne dinamičke nizove. Umjesto toga, matricu možemo zamisliti kao niz redova (vidi sliku), gdje memoriju za svaki red alociramo odvojeno.



   Dakle, prvo trebamo alocirati memoriju za pointere koji pokazuju na redove. Njih ima onoliko koliko ima redova, dakle `matrix = new char*[rows];`, a zatim, za svaki red alocirati memoriju za taj red:

```
for(int i = 0; i < rows; i++)
   matrix[i] = new char[cols];
```

   Po tipu, svaki matrix[i] je pointer, pa je matrix dvostuki pointer: `char** matrix;`

   Sve skupa izgledalo bi ovako:

```
char** matrix;
matrix = new char*[rows];
for(int i = 0; i < rows; i++)
   matrix[i] = new char[cols];
```

   Dalje radimo sa matricom kao sa običnim dvodimenzionalnim nizom. Naprimjer, ako želimo unositi članove matrice, napravili bi to ovako:

```
for(int i = 0; i < rows; i++)
   for(int j = 0; j < cols; j++)
      cin >> matrix[i][j];
```

   Oslobađanje memorije ide obratno od alokacije. Prvo oslobodimo memoriju koju zauzimaju redovi matrice, a onda pointere koji pokazuju na te redove.

```
for(int i = 0; i < rows; i++)
        delete [] matrix[i];
     delete [] matrix;
```

2. **Alokacija pomoću vektora vektora.** Vektori se također alociraju na heapu, tako da u zadatku možemo koristiti i vektore.

   (a) Unaprijed zadana veličina matrice (bolje u našem slučaju).

```
vector<vector<char> > matrix(rows, vector<char>(cols));
```

Nakon toga unosimo ili pridružujemo vrijednosti članovima.

```
for (int i = 0; i < rows; ++i)
{
   for(int j = 0; j < cols; ++j)
   {
      cin >> matrix[i][j];
   }
}
```

(b) Vektor "gradimo" član po član

```
vector<vector<char>> matrix;

for (int i = 0; i < rows; ++i)
{
   vector<char> v;
   for(int j = 0; j < cols; ++j)
   {
      char a;
      cin >> a;
      v.push_back(a);
   }
   matrix.push_back(v);
}
```

3. Još jedan način je korištenjem jednodimenzionalnih nizova.

```
matrix = new char[rows*cols];
```

Iako efikasan, manje je pregledan način i ne preporuča se. Ono što je dobro kod tog načina, je da se memorija alocira uzastopno za sve članove i to red po red. Naime, memorija je jednodimenzionalna i u svim prethodnim metodama smo sigurni da će se elementi jednog reda nalaziti na uzastopnim memorijskim lokacijama, ali nismo sigurni da će redovi ići jedan za drugim. U ovom slučaju sigurni smo da će se cijela matrica nalaziti na uzastopnim mjestima u memoriji.

```
row 0       row 1       row 2
|           |           |           |
+---+---+---+---+---+---+---+---+---+---+---+---+
| A | B | C | D | E | F | G | H | I | J | K | L |
+---+---+---+---+---+---+---+---+---+---+---+---+
|<-- columns -->|<-- columns -->|<-- columns -->|
```

Ako zamislimo da je ovo matrica koju gledamo ovako:

```
   0   1   2   3
  +---+---+---+---+
0 | A | B | C | D |
+---+---+---+---+
1 | E | F | G | H |
+---+---+---+---+
2 | I | J | K | L |
+---+---+---+---+
```

onda bi se vrijednosti zapisale na ovaj način `M[i][j] (2-dim)<=> matrix[ i * cols + j ] (1-dim)`

U donjem primjeru element `M[2][3]` odgovarao bi elementu

`matrix[2 * cols + 3]`.

Uočite taj element na slici:

```
row 0       row 1       row 2
+---+---+---+---+---+---+---+---+---+---+---+---+
| A | B | C | D | E | F | G | H | I | J | K | L |
+---+---+---+---+---+---+---+---+---+---+---+---+
|<-     i * cols      -->|<- j  ->|
```

Unos sa tastature bi sada mogli napisati ovako:

```
for(int i = 0; i < rows; ++i)
   for (int j = 0; j < cols; ++j)
      cin >> matrix[i * cols + j];
```

Niz naravno treba dealocirati (u destruktoru).

U našem zadatku osim punjenja matrice tako da je ispunimo znakom praznine trebate još i na rubovima postavit elemente na neki `char`. Nadam se da to nije problem :)

Razmislite kako bi implementirali trodimenzionalnu matricu.