Динамично заделена матрица. init, print, clean
Идеята за “назъбена” матрица.

Увод в работата със символни низове

- **ASCII recap**

- **Защо използваме символни низове?**

- Типове:
  - Статичен масив.

  Пример:
  const char animal = { ‘D’ , ‘o’,’ ’g’ , ‘\0’ };
  const char* animal = “Bird”;
  const char * animals[] = { "cat", "dog", "cow" };

  - Динамичен масив.

  Кога е подходящо да използваме всеки един?
  Как изглежда всеки един в паметта?

- Символ за край на низа : **‘\0’** , така наречената терминиращата 0.

  Защо ни трябва символ за край на низа?

  Важно!!!
  char * name = “Tosho” ⇔ char name[] = {‘T’,’o’,’s’,’h’,’o’,**’\0’**};
  но с първото сочим в памет, която не можем да модифицираме, докато второто заделя паметта на стека.

  пробвайте char name[] = {‘T’,’o’,’s’,’h’,’o’};
  и след това std::cout << name;

- **Четене на символен низ от конзолата**
  - std::cin
  - std::cin.getline()

  Пример:
  std::cin.getline(<буфер> , <размер на буфера>);
  std::cin.getline(<буфер> , <размер на буфера>, <символ>);

  *където <символ> е специален символ, до който искаме да спрем четенето

За повече информация:
http://www.cplusplus.com/reference/iostream/cin/?kw=cin
http://www.cplusplus.com/reference/istream/istream/getline/
http://www.cplusplus.com/reference/istream/istream/get/
http://www.cplusplus.com/reference/istream/istream/ignore/

- **Писане на символен низ на конзолата**
  - std::cout
  - std::cerr

  Кога ползваме всеки от потоците?

Подаване на символен низ на функция.
Няма нужда да влачим “променлива за размера”.Защо ?

стандартните функции за работа с низове
и техни примерни реализации

- основните : strlen, strcmp, strcpy, strcat

- разлика с вресиите strncpy, strncat, strncmp

- кои други вършат работа? strtok, strstr, memset, memcpy

*Ползваме си тези от библиотеката “string.h”,но можем да си ги напишем, защото знаем как работят :
http://www.cplusplus.com/reference/cstring/