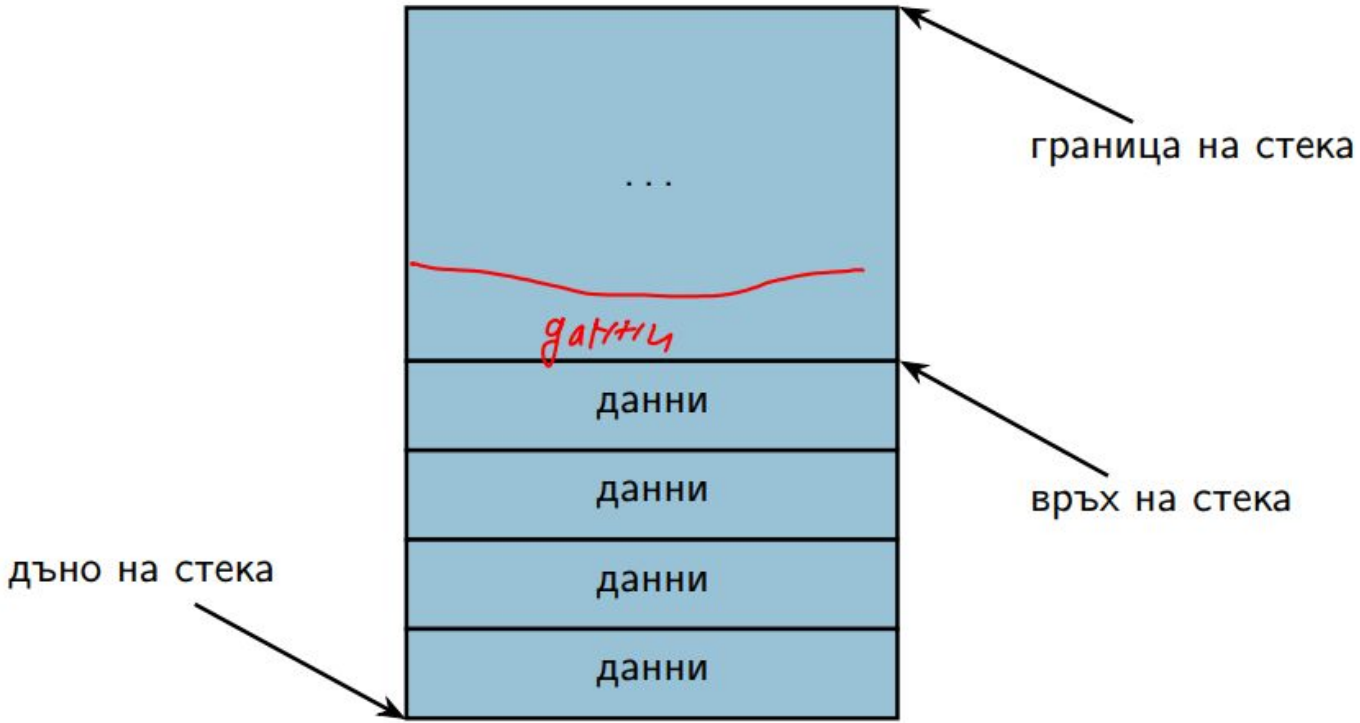
Динамична памет

УП практикум, 2ра група
Богомил Стоянов
Виолета Кастрева

Програмен стек

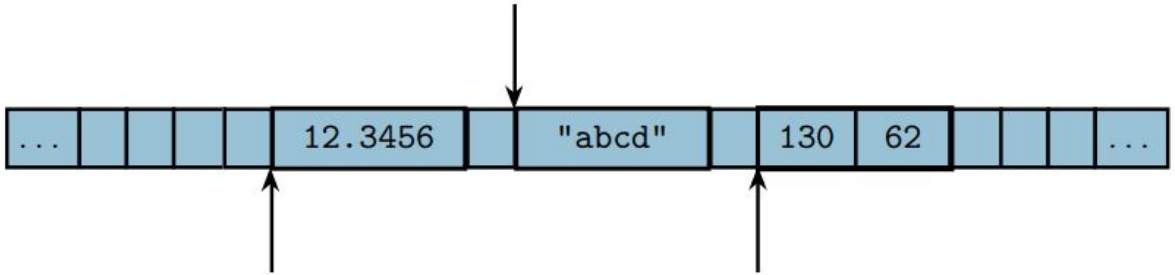


Програмен стек

- Паметта се заделя в момента на дефиниция
- Всеки заделен блок памет носи името на променливата
- Паметта се освобождава при изход от блока (или функцията), в който е дефинирана променливата
- Последно заделената памет се освобождава първа
- Програмистът няма контрол над управлението на паметта.
- Количеството заделена памет до голяма степен е определено по време на компилация
- **Статично заделяне на памет**

Област за динамична памет (heap)

- Динамичната памет може да бъде заделена и освободена по всяко време на изпълнение на програмата
- Областта за динамична памет е набор от свободни блокове памет
- Програмата може да заяви блок с произволна големина



Как да заделяме динамична памет?

- **new** : заделя памет в heap-а и я инициализира.
- **delete** : освобождава памет от heap-а, за да предотврати “изтичане на памет” (memory leak).

```
1 int* ptr = new int; // Allocation
2 *ptr = 5; // Use
3 delete ptr; // Deallocation
4
```

Защо е толкова важно да нямаме memory leaks?

Причините могат да запълнят цяла презентация: излишно запълване на памет, забавяния на програмата, непредвидено поведение и т.н.

По-интересни причини:

- Ще ви се взимат точки на контролни ако не го правите
- На един колега му се изключи компютъра (по време на изпит) заради пропуснати delete

Работа с динамични масиви

- Създаване на масив, чийто размер се определя по време на компилация

```
1 int* arr = new int[10]; // Dynamic array allocation
2 // Initializing array elements
3 for(int i = 0; i < 10; ++i) {
4     arr[i] = i;
5 }
6 delete[] arr; // Array deallocation
7
8
```

Пример с матрица:

```cpp
#include <iostream>
using namespace std;

int main() {
    int rows = 3, cols = 3;
    int** matrix = new int*[rows]; // Allocate array of pointers

    // Allocate and initialize each row
    int count = 0;
    for (int i = 0; i < rows; ++i) {
        matrix[i] = new int[cols];
        for (int j = 0; j < cols; ++j) {
            matrix[i][j] = ++count; // Assign sequential numbers
        }
    }

    // Deallocate the matrix
    for (int i = 0; i < rows; ++i) {
        delete[] matrix[i]; // Deallocate each row
    }
    delete[] matrix; // Deallocate array of pointers

    return 0;
}
```

Подаване към функция:

```
int sumMatrix(int** matrix, int rows, int cols) {
    int sum = 0;
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            sum += matrix[i][j];
        }
    }
    return sum;
}

int main() {
    int rows = 2, cols = 3;
    int** matrix = new int*[rows];
    // Initialize matrix with values...
    int totalSum = sumMatrix(matrix, rows, cols);
    // Cleanup
    for (int i = 0; i < rows; ++i) {
        delete[] matrix[i];
    }
    delete[] matrix;

    return 0;
}
```