Динамично заделяне на памет

гл.ас. д-р. Нора Ангелова

Динамични обекти

- Памет – програмен стек, област на динамичните данни (хийп):
  - Програмен стек – кратковременна памет. Елементите му се наричат стекови рамки.
  - Хийп – не се свързва с имена на променливи. Работи се с указатели.

Динамични обекти

Структури от данни, за които операциите включване (добавяне) и изключване на елемент не са допустими, се наричат **статични**, в противен случай – **динамични.**

Динамични обекти

- Заделянето на памет по време на компилация се нарича **статично заделяне на памет**.
- Заделянето на памет по време на изпълнение на програмата се нарича **динамично заделяне на памет**.

Динамични обекти

Динамични обекти:

- Създаване на динамични обекти – оператор new.
- Разрушаване на динамични обекти – оператор delete.

** Заделената по този начин памет остава свързана със съответната променлива докато не се освободи явно от програмиста.*

Динамични обекти

Оператор new:

- Използва се за създаване на динамични обекти.
- Заделя в хийпа необходимата памет и връща указател към нея.
- Пази се докато е необходимо.

```
new <име_на_тип>;
new <име_на_тип> [ size ];
new  <име_на_тип> (<инициализация>);
```

<име_на_тип> - име на стандартен тип или име на клас.

size – израз с произволна сложност, която може да се преобразува до цял с положителна стойност. Показва броя на компонентите от типа, за които да се задели памет в хийпа и се нарича **размерност.**

<инициализация> - израз от тип <име_на_тип> или инициализация на обект според синтаксиса на конструктора на класа, ако <име_на_тип> е име на клас.

Динамични обекти

Оператор new:

- Ако няма достатъчно място в хийпа, операторът new връща NULL или изключение.

Пример:

```
int* dynamicArr = new int[10];
int* dynamicVar = new int(2+5*5);
int** arrPointers = new int*[5];
```

** Проверка дали паметта е заделена*

Динамични обекти

Оператор delete:

- Разрушава обекта, адресиран от указателя и паметта, която заема този обект се освобождава.

```
delete <указател_към_динамичен_обект>;
delete [] <указател_към_динамичен_обект>;
delete [size] <указател_към_динамичен_обект>;
```

<указател_към_динамичен_обект> - указател към диначен обект, създаден чрез оператора new.

Пример:

```
int * dynamicVar = new int(2+5*5);
delete dynamicArr;
```

Динамични обекти

Оператор delete:

- Разрушаване на масив

Пример:

```
int * dynamicVar = new int[10];
delete [] dynamicArr;
```

** Ако масивът съдържа в себе си указатели, които адресират динамично заделена памет, първо масивът трябва да бъде обходен и за всеки негов елемент да бъде извикан операторът delete.*

Динамични обекти

Задача
Да се създаде масив от указатели към цели числа (размерът се определя от променлива).

Нека стойностите да съответстват на индекса на елементите.

Да се изведат стойностите на екрана.

Динамични обекти

```
int size = 10;

// Създаване
int** arr = new int*[size];
if (!arr) {
  cout << "Not enough memory!\n";
  return 1;
}

for(int i = 0; i < size; i++) {
  arr[i] = new int;

  if (!arr[i]) {
    cout << "Not enough memory!\n";
    return 1;
  }
  *arr[i] = i;
}

// Извеждане
for(int i = 0; i < size; i++) {
  cout << *arr[i] << " ";
}

// Разрушаване
for(int i = 0; i < size; i++) {
  delete arr[i];
}
delete [] arr;
```

Динамични обекти

```
#include <cassert>
```

```
assert(<твърдение>)
```

Ако е истина, изпълнението на програмата продължава, а ако не е истина, изпълнението на програмата завършва със съобщение за грешка, включващо мястото на възникване на грешката в кода (име на файл и номер на ред), както и самия текст на условието, което не е било изпълнено.

Динамични обекти

```
int** arr = new int*[size];

if (!arr) {
  cout << "Not enough memory!\n";
  return 1;
}

-------------------------------------

assert(arr != NULL);
```

Край