

Динамична памет

Трифон Трифонов

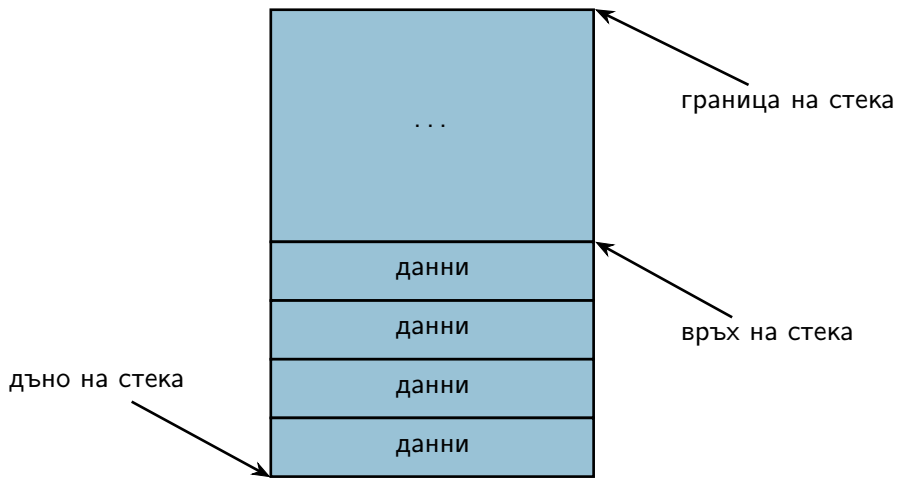
Обектно-ориентирано програмиране,
спец. Компютърни науки, 1 поток,
2018/19 г.

6 март 2019 г.

Схема на програмната памет



Програмен стек



Свойства на програмния стек

- Паметта се заделя в момента на дефиниция

Свойства на програмния стек

- Паметта се заделя в момента на дефиниция
- Всеки заделен блок памет носи името на променливата

Свойства на програмния 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

Свойства на програмния 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

Свойства на програмния 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

Свойства на програмния 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
 - Паметта не може да се освободи по-рано (преди края на блока)

Свойства на програмния 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
 - Паметта не може да се освободи по-рано (преди края на блока)
 - Паметта не може да се запази за по-дълго (след края на блока)

Свойства на програмния 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
 - Паметта не може да се освободи по-рано (преди края на блока)
 - Паметта не може да се запази за по-дълго (след края на блока)
- Количеството заделена памет до голяма степен е определено по време на компилация

Свойства на програмния 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
 - Паметта не може да се освободи по-рано (преди края на блока)
 - Паметта не може да се запази за по-дълго (след края на блока)
- Количеството заделена памет до голяма степен е определено по време на компилация
 - При какви случаи заделената памет може да варира по време на изпълнение?

Област за динамична памет (heap)

- Динамичната памет може да бъде заделена и освободена по всяко време на изпълнение на програмата

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

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

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
- Операционната система се грижи за управлението на динамичната 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
- Операционната система се грижи за управлението на динамичната памет
 - поддържа “карта” кои клетки са свободни и кои заети

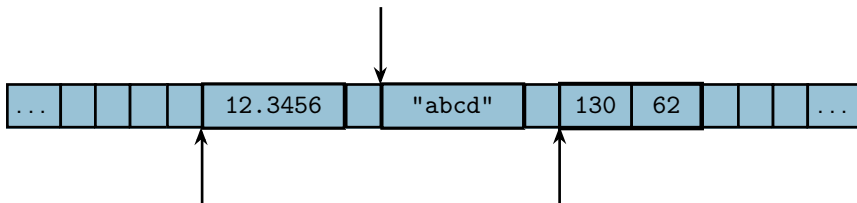
Област за динамична памет (heap)

- 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
- Операционната система се грижи за управлението на динамичната памет
 - поддържа “карта” кои клетки са свободни и кои заети
 - контролира коя част от паметта от коя програма се използва (защитен режим)

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
- Операционната система се грижи за управлението на динамичната памет
 - поддържа “карта” кои клетки са свободни и кои заети
 - контролира коя част от паметта от коя програма се използва (защитен режим)
 - позволява използването на външни носители (виртуална памет)

Схема на динамичната памет



Заделяне на динамична памет

- Заделянето на динамична памет става с операциите `new` и `new []`

Заделяне на динамична памет

- Заделянето на динамична памет става с операциите **new** и **new []**
- **new** <тип> — заделя блок от памет за една променлива от <тип>

Заделяне на динамична памет

- Заделянето на динамична памет става с операциите **new** и **new []**
- **new** <тип> — заделя блок от памет за една променлива от <тип>
- **new** <тип> [<брой>] — заделя блок от памет за масив от <брой> елемента от <тип>

Заделяне на динамична памет

- Заделянето на динамична памет става с операциите **new** и **new []**
- **new** <тип> — заделя блок от памет за една променлива от <тип>
- **new** <тип> [<брой>] — заделя блок от памет за масив от <брой> елемента от <тип>
- **new** <тип> (<инициализация>) — заделя блок от памет за една променлива от <тип> и я инициализира със зададените един или повече параметри

```
int a = 2;  
new int(2);  
new Rational(1, 2);
```


Заделяне на динамична памет

```
int *p = new int[10];
```

- Заделянето на динамична памет става с операциите **new** и **new []**
- **new** <тип> — заделя блок от памет за една променлива от <тип>
- **new** <тип> [<брой>] — заделя блок от памет за масив от <брой> елемента от <тип>
- **new** <тип> (<инициализация>) — заделя блок от памет за една променлива от <тип> и я инициализира със зададените един или повече параметри
- връща указател <тип>* към новозаделения блок

```
int *p = new int(2);
```

~~cout << p[3];~~

Примери за заделяне

- `int* p = new int;`



Примери за заделяне

- `int* p = new int;`



- `float* q = new float(1.23);`



Примери за заделяне

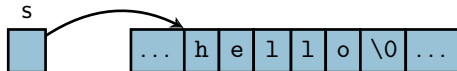
- `int* p = new int;`



- `float* q = new float(1.23);`



- `char* s = new char[6];`
`strcpy(s, "hello");`



Примери за заделяне

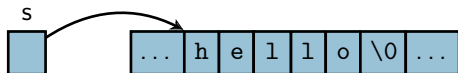
- `int* p = new int;`



- `float* q = new float(1.23);`



- `char* s = new char[6];`
`strcpy(s, "hello");`



- `char** ss = new char*(s);`



*char *s1 = s;*

Освобождаване на памет

- Динамична памет се освобождава с операциите `delete` и `delete[]`

Освобождаване на памет

- Динамична памет се освобождава с операциите `delete` и `delete[]`
- `delete` <указател> — освобождава блок от памет с начало, сочено от <указател>

Освобождаване на памет

- Динамична памет се освобождава с операциите `delete` и `delete[]`
- `delete` <указател> — освобождава блок от памет с начало, сочено от <указател>
- `delete[]` <указател> — освобождава блок от памет, съдържащ масив от обекти, първият от които е сочен от <указател>

Освобождаване на памет

- Динамична памет се освобождава с операциите `delete` и `delete[]`
- `delete` <указател> — освобождава блок от памет с начало, сочено от <указател>
- `delete[]` <указател> — освобождава блок от памет, съдържащ масив от обекти, първият от които е сочен от <указател>
 - указването на число в скобите не е нужно, понеже операционната система “знае” колко е голям заделения блок

Ограничения при освобождаването на памет

- На `delete` може да се подаде само указател, върнат от `new`

Ограничения при освобождаването на памет

- На `delete` може да се подаде само указател, върнат от `new`
- Не е позволено освобождаването на памет в програмния стек

Ограничения при освобождаването на памет

- На `delete` може да се подаде само указател, върнат от `new`
- Не е позволено освобождаването на памет в програмния стек
 - `int x; int* p = &x; ... delete p;`

Ограничения при освобождаването на памет

- На `delete` може да се подаде само указател, върнат от `new`
- Не е позволено освобождаването на памет в програмния стек
 - `int x; int* p = &x; ... delete p;`
- Не е позволено частично освобождаване на памет

Ограничения при освобождаването на памет

- На `delete` може да се подаде само указател, върнат от `new`
- Не е позволено освобождаването на памет в програмния стек
 - `int x; int* p = &x; ... delete p;`
- Не е позволено частично освобождаване на памет
 - `int* a = new int[10]; ... delete (a+2);`

Ограничения при освобождаването на памет

- На `delete` може да се подаде само указател, върнат от `new`
- Не е позволено освобождаването на памет в програмния стек
 - `int x; int* p = &x; ... delete p;`
- Не е позволено частично освобождаване на памет
 - `int* a = new int[10]; ... delete (a+2);`
- Не е позволено използването на памет след като е освободена

```
int * p = new [10]; ----- delete[] p;
... cout << p[2];
```

Ограничения при освобождаването на памет

- На `delete` може да се подаде само указател, върнат от `new`
- Не е позволено освобождаването на памет в програмния стек
 - `int x; int* p = &x; ... delete p;`
- Не е позволено частично освобождаване на памет
 - `int* a = new int[10]; ... delete (a+2);`
- Не е позволено използването на памет след като е освободена
- Не е позволено повторното освобождаване на една и съща памет

Ограничения при освобождаването на памет

- На `delete` може да се подаде само указател, върнат от `new`
- Не е позволено освобождаването на памет в програмния стек
 - `int x; int* p = &x; ... delete p;`
- Не е позволено частично освобождаване на памет
 - `int* a = new int[10]; ... delete (a+2);`
- Не е позволено използването на памет след като е освободена
- Не е позволено повторното освобождаване на една и съща памет
 - `int* p = new int[5], *q = p; delete p; q[1] = 5; delete q;`



Задачи за динамична памет

- 1 Да се напише програма, която въвежда няколко положителни дробни числа в динамичната памет и намира средното им аритметично

Задачи за динамична памет

- 1 Да се напише програма, която въвежда няколко положителни дробни числа в динамичната памет и намира средното им аритметично
- 2 Да се напише програма, която създава матрица от числа в динамичната памет и я транспонира



$a[2][3]$?

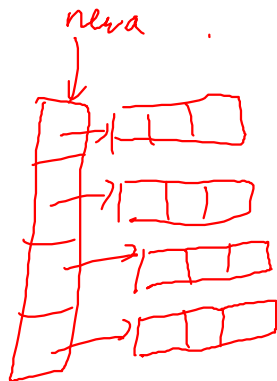
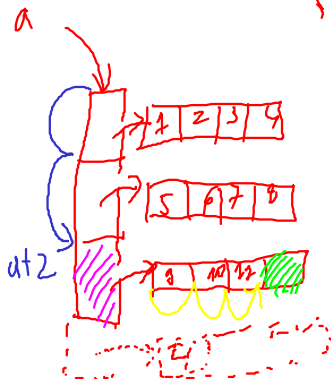
double $a[2][3]$;

$a[1]$ → указатель



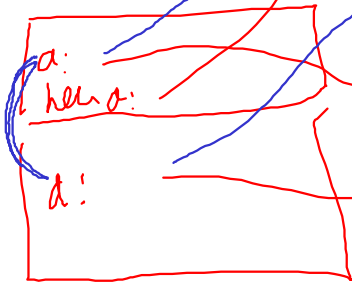
← double ($*[3]$)
мы не знаем с 3
ед.

$$a[2][3] \leftrightarrow * (* (a+2) + 3)$$



transposeMatrix

testTranspose



Особености на динамичната памет

- Програмистът има контрол над заделянето на памет

Особености на динамичната памет

- Програмистът има контрол над заделянето на памет
- Програмистът носи отговорност за правилната работа с динамичната памет

Особености на динамичната памет

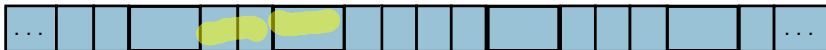
- Програмистът има контрол над заделянето на памет
- Програмистът носи отговорност за правилната работа с динамичната памет
- Заделената динамична памет остава непокътната до освобождаването ѝ с `delete` или до завършване на програмата

Особености на динамичната памет

- Програмистът има контрол над заделянето на памет
- Програмистът носи отговорност за правилната работа с динамичната памет
- Заделената динамична памет остава непокътната до освобождаването ѝ с `delete` или до завършване на програмата
- След приключване на програмата, цялата заделена от нея памет се освобождава от операционната система

Особености на динамичната памет

- Програмистът има контрол над заделянето на памет
- Програмистът носи отговорност за правилната работа с динамичната памет
- Заделената динамична памет остава непокътната до освобождаването ѝ с `delete` или до завършване на програмата
- След приключване на програмата, цялата заделена от нея памет се освобождава от операционната система
- Честото заделяне и освобождаване на малки блокове памет води до **фрагментация**



Особености на динамичната памет

- Програмистът има контрол над заделянето на памет
- Програмистът носи отговорност за правилната работа с динамичната памет
- Заделената динамична памет остава непокътната до освобождаването ѝ с `delete` или до завършване на програмата
- След приключване на програмата, цялата заделена от нея памет се освобождава от операционната система
- Честото заделяне и освобождаване на малки блокове памет води до **фрагментация**



- Динамично заделените блокове памет не се свързват с имена

Грешки при работа с динамична памет

- Работа с указател към незаделена или освободена памет

Грешки при работа с динамична памет

- Работа с указател към незаделена или освободена памет
- Освобождаване на непозволена памет

Грешки при работа с динамична памет

- Работа с указател към незаделена или освободена памет
- Освобождаване на непозволена памет
- “Загубване” на указател към заделена памет

Грешки при работа с динамична памет

- Работа с указател към незаделена или освободена памет
- Освобождаване на непозволена памет
- “Загубване” на указател към заделена памет
- Неосвобождаване на неизползвана памет

Грешки при работа с динамична памет

- Работа с указател към незаделена или освободена памет
- Освобождаване на непозволена памет
- “Загубване” на указател към заделена памет
- Неосвобождаване на неизползвана памет
- Изтичане на памет (memory leak)