Увод в програмирането

Лекция 9:

Стекова и динамична памет

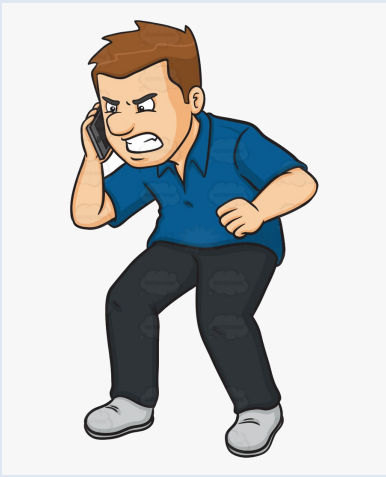
(Само за **Информатика и ИС**)

Стек

- Колекция от елементи
- Основни операции:
  - Добавяне на елемент
  - Извличане на последно добавения елемент
  - Достъп до елемента на върха
- Изучава се в курса по СДП

Ако все още не е ясно какво е стек...

- Опитайте се да изкарате най-вътрешната кола :)

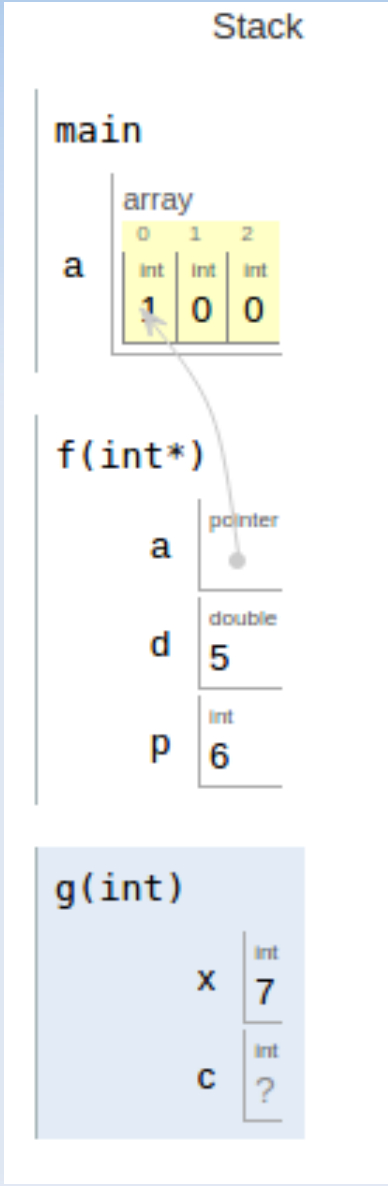


Стекова памет

- Паметта, използвана от локалните променливи на функциите, е организирана като стек
- Елементи - стекови рамки

Визуална демонстрация в pythontutor.com

```
void g(int x)
{
   int c;
   c = x;
}
void f(int *a)
{
   double d = 5;
   if (d > 0)
   {
      int p = 6;
      g(p + *a);
      p++;
   }
}
int main()
{
   int a[3] = {1};
   f(a);
   g(2);
   return 0;
}
```



Особености

- Нямаме контрол над управлението на паметта
  - Да си припомним: област на видимост на променлива
  - Паметта не може да се освободи по-рано или по-късно
- Количеството заделена памет до голяма степен е определено по време на компилация

Област за динамична памет (heap)

- Може да бъде заделена и освободена по всяко време на изпълнение на програмата
- Програмата може да заяви блок с произволна големина (напр. въведен от потребителя)
- Операционната система (ОС):
  - поддържа карта кои клетки са свободни и кои - не
  - контролира коя част от паметта от коя програма се използва
- Оператори new и delete

Заделяне на динамична памет

- **`new <тип>`** заделя блок от памет за една променлива от дадения тип

```
int *p = new int;
```

- **`new <тип>[<брой>]`** заделя блок от памет за масив с дадения брой елементи

```
int n; cin >> n;
double *arr = new double[n];
arr[0] = 3.14;
```

- Операторът връща указател <тип>* към новозаделения блок
- или грешка, ако операцията не може да бъде изпълнена (няма достатъчно памет)

Освобождаване на динамична памет

- В C++ трябва изрично да освободим заделената с new памет
- **`delete <указател>`** освобождава блок от памет с начало, сочено от <указател>

```
int *p = new int;
delete p;  // освобождава sizeof(int) байта
```

- **`delete [[<брой>]]<указател>`** освобождава блок от памет, съдържащ масив от <брой> елементи
  - Указването на броя не е задължително, понеже ОС знае колко е голям заделеният блок

Особености

- На delete може да се подаде само адрес, върнат от new
  - `int a; int* p = &a;`
    `~~delete p;~~ // a е в стека`
  - `int *q = new int[5];`
    `int *r = &q[2] - 2;`
    `delete [] r; // позволено ли е?`
    `~~delete [] (q + 1);~~`
- `int *s = new int;`
  `delete s;`
  `~~*s = 5;~~ // паметта вече е освободена`
  `~~delete s;~~ // паметта вече е освободена`

Memory leak

- Има една най-честа грешка при работа с динамична памет:
- Неосвобождаването на памет, заделена с new
  - Забравили сме да напишем delete, когато паметта вече не е необходима
  - Загубили сме указателя, който сочи към паметта
- Получава се memory leak

Задачи

- Да се напише програма, която по дадено цяло положително число *n* създава масив с точно *n* реални числа, след което намира средното им аритметично
- Да се напише функция, която по дадена правоъгълна матрица от реални числа намира сумата на елементите, които се намират на редове с четен индекс
- Да се напише функция, която по дадено число n връща низ с необходимата дължина, съдържащ n повторения на думата „love“, напр. „lovelovelove“
- Визуализация на решенията в pythontutor.com

Допълнителен материал

- Garbage Collector
  - В Java и други езици не се налага ръчно да освобождаваме заделената с new памет
- Stack trace

Край на презентацията