Ponovitev C-ja:

- header file - vključimo funkcije, razrede iz druge datoteke
- overloadanje funkcij - pomembno samo ime funkcij, ne pa tudi return type ali tipi/število argumentov
- return 0 - program se je pravilno zaključil, v bash je 0 "true"; `echo $?`
- error kode - npr. 139 segmentation fault
- char = 8 bytov, lahko ga beremo kot številko

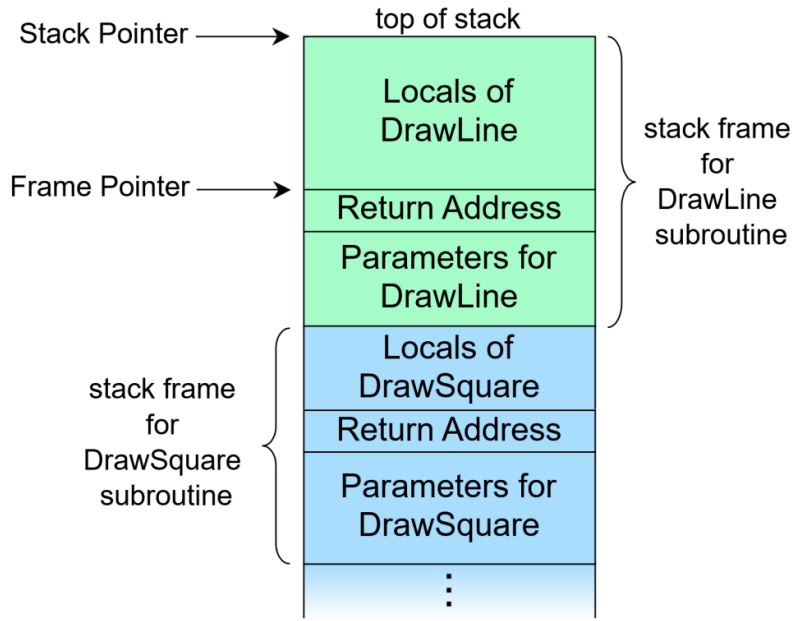
```
char *a = "fri";
```

- `\0` za konec stringa (string je v bistvu char pinter)
- `char **argv` je tabela char pointerjev, vrednosti stringov niso nujno zaporedno v pomnilniku; dostop: `argv[i]`, za konec posameznega stringa
- `char **env` - dobimo še okoljske spremenljivke, v bash `env`
- da naredimo nov proces - `fork` iz enega procesa naredi kopijo, je sistemski klic
- na fork se potem naredi kliče `exec`
- starševski proces vmes dela `wait` in ko otrok umre, starš prevzame njegov izhodni status
- %p je pointer (v bistvu hex)
- reference v Javi so v bistvu samo pointerji na pointerje, ker potem lahko dela garbage collection - si zapomnimo koliko pointerjev kaže na neko strukturo in ko noben več ne kaže gor, ga lahko zbrišemo, štejemo koliko pointerjev kaže na eno strukturo
- v C moramo sami delati garbage collection
- malloc sistemski klic:
  - če je vse v redu vrne 0, če ni v redu vrne -1, lahko tudi vrne nekaj večje od 0 (naslov, ki smo ga rezervirali)
  - lahko faila
  - fork vrne PID procesa (staršu da PID otroka, otroku pa 0), če je failal, vrne -1

```c
int izpisi_nekaj(char *str)
{
    int a = 5 + 37;
    printf("%s\n", str);
    return a;
}

int main()
{
    int b = izpisi_nekaj("hej");
    exit(b);
}
```

- "hej" se shrani v register RAX, zato ne rabi iti na stack
- rbp = base pointer (frame pointer)
- rsp = stack pointer



- registri na x86 arhitekturi:

identifiers to access registers and parts thereof

Register	Accumulator	Base	Counter	Stack Pointer	Stack Base Pointer	Destination	Source	Data
64-bit	RAX	RBX	RCX	RSP	RBP	RDI	RSI	RDX
32-bit	EAX	EBX	ECX	ESP	EBP	EDI	ESI	EDX
16-bit	AX	BX	CX	SP	BP	DI	SI	DX
8-bit	AH, AL	BH, BL	CH, CL	SPL	BPL	DIL	SIL	DH, DL

- RIP = instruction pointer
- klic maina-a je zelo podoben klicu poljubne funkcije - uporablja iste registre in samo stare vrednosti shrani na stack in da nove vrednosti v registre

- exit status je večja stvar kot exit code
- lahko vrne 2^8 različnih exit code-ov
- če hočemo nek izhodni rezultat prebrati, ga dobimo v enem registru (RAX)
- exit vzame exit code kot parameter iz RAX
- lahko delamo pointer arithmetic
- `void*` je pointer na "nekaj", velik 1 byte