



Financira  
Evropska unija  
NextGenerationEU



THE RECOVERY  
AND RESILIENCE  
PLAN

# VARNOST PROGRAMOV



Predavanja #1  
Matevž Pesek



# DANAŠNJE TEME

Obnovitev Basha

Razlaga okolja za vaje

# Ponovitev osnov

## Programski jezik C

- Ker ga potrebujemo
- Ker je povsod nekje globoko še kaj spisano v C-ju
- Ker je C potreben za razumevanje delovanja višjenivojskih jezikov

## Bash/CLI

- Povsod na voljo
- Navadno edini način za komunikacijo z oddaljenim računalnikom
- Ker omogoča manipulacijo (vsega) na nivoju sistema



# C

The best/worst language in the world :)



# IZKUŠNJE?

Dvignite roke



# Tipičen program

- Header datoteke (funkcije)
- Funkcija main (zakaj ne metoda?)
  - int main / void main?
  - Argumenti?
- Return [int]

```
#include <stdio.h>

int main() {
    //void main()
    //int main (int argc, char *argv[])
    // int main (int argc, char **argv, char *envp[])
    printf("Hello, world!");
    return 0;
}
```

# Kje so podatki

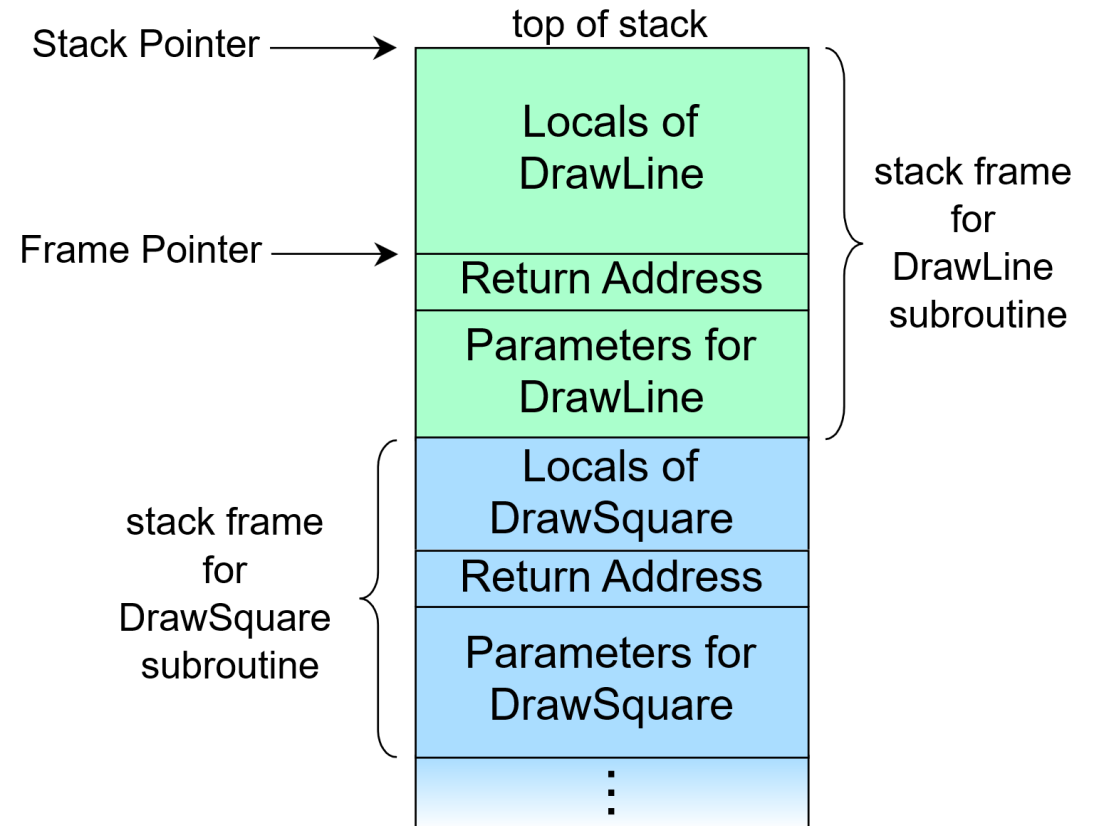
- Kazalci (pointerji!)
  - Stack pointer, Frame pointer

Kaj se zgodi, ko pokličem funkcijo?

```
#include <stdio.h>

void izpisi_nekaj(char *besedilo) {
    printf("%s", besedilo);
}

int main {
    izpisi_nekaj("Hello, world!")
    return 0;
}
```



By R. S. Shaw - Own work, Public Domain,  
<https://commons.wikimedia.org/w/index.php?curid=1956587>

# Kazalci!

```
int myAge = 43; // an int variable

printf("%d", myAge); // Outputs the value of myAge (43)
printf("%p", &myAge); // Outputs the memory address of myAge (0x7ffe5367e044)

int* ptr = &myAge; // Pointer declaration
// Reference: Output the memory address of myAge with the pointer
// (0x7ffe5367e044)
printf("%p\n", ptr);

// Dereference: Output the value of myAge with the pointer (43)
printf("%d\n", *ptr);
```

[https://www.w3schools.com/c/c\\_pointers.php](https://www.w3schools.com/c/c_pointers.php)



# Kazalčna aritmetika

```
int n = 10;
int *a;
a = &n;
printf("current address = %x \n",a);
a = a+1;
printf("next address = %x \n",a);
```

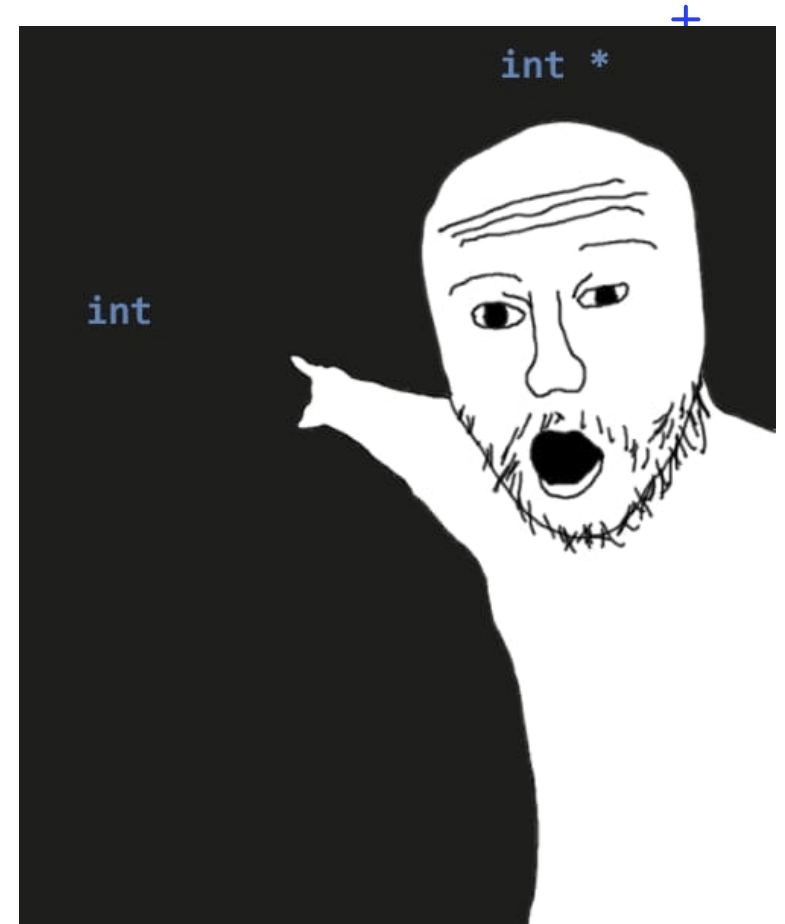
- Kakšen bo naslednji naslov?
- $\text{Next Location} = \text{Current Location} + i * \text{size\_of}(\text{data type})$

# Kazalčna aritmetika #2

```
#include <stdio.h>

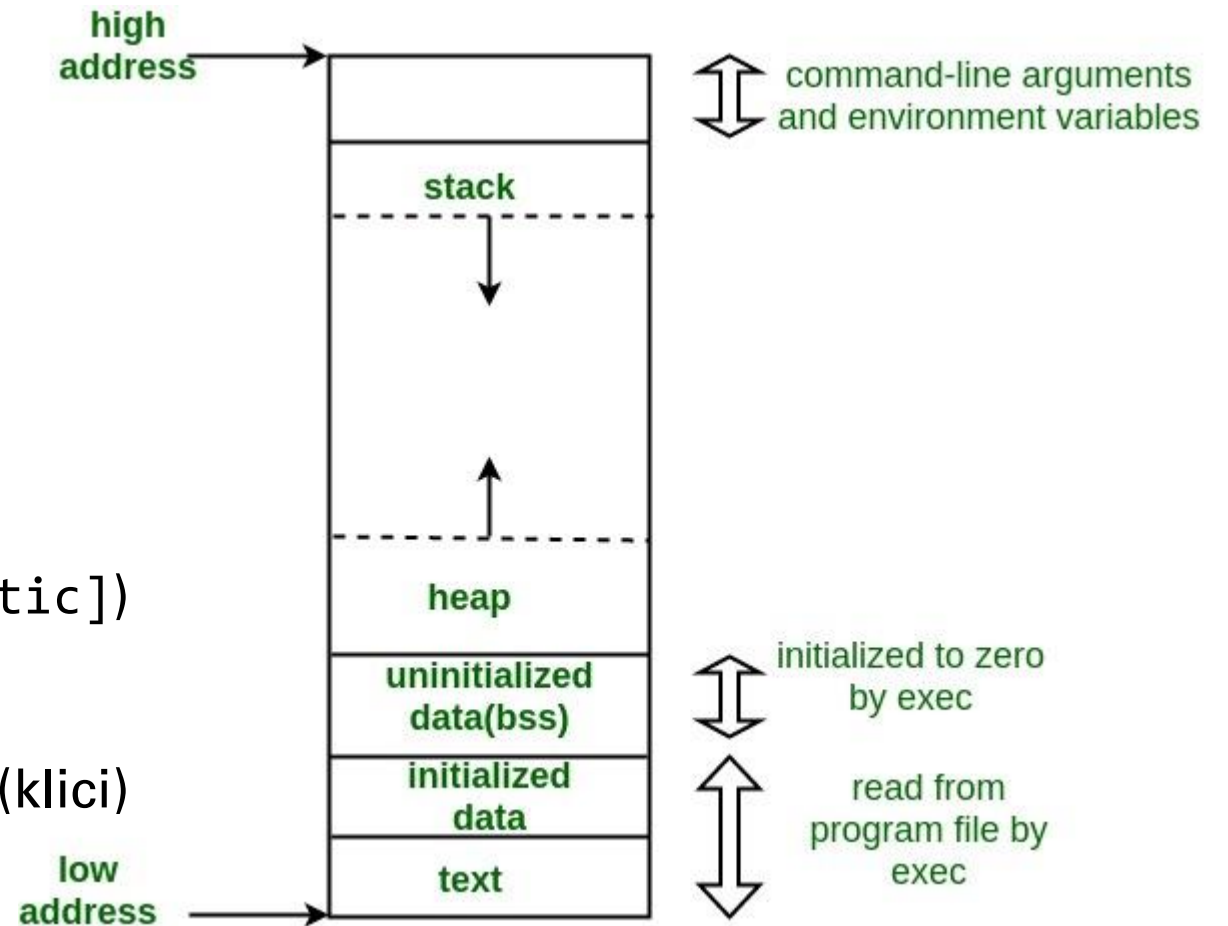
int main() {
    printf("-0.5" + 1);
}
```

- Kaj izpiše proces?
- Zakaj?



# Sklad/kopica?

- .text: koda v strojnem jeziku
- .data: inicializirani podatki (globalne spremenljivke [const])
- .bss: neinicializirani podatki (npr. [static])
- Heap: kopica -> malloc
- Stack: sklad -> lokalne spremenljivke, (klici) funkcij

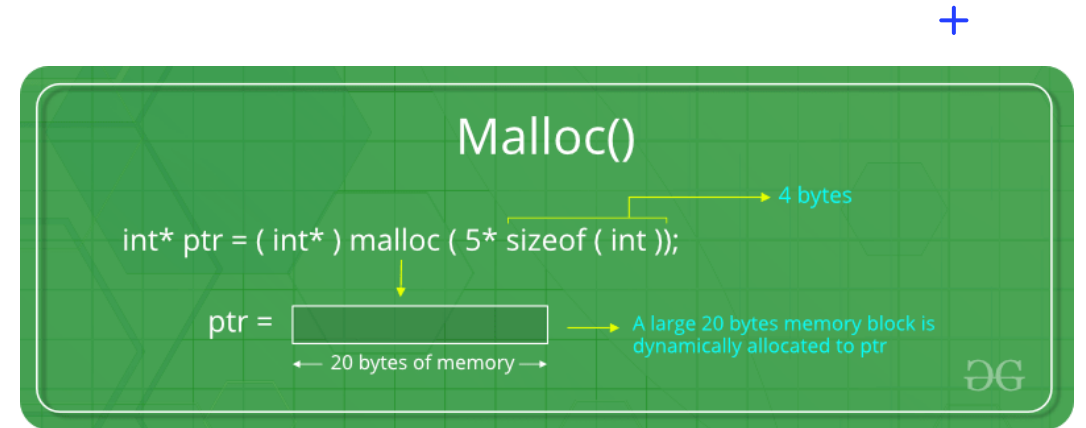


# malloc

- Dinamična alokacija pomnilnika

- `ptr = (int*) malloc(100 * sizeof(int));`
- `if (ptr == NULL) -> problem!`

```
printf("Moji elementi so: ");  
for (i = 0; i < n; ++i) {  
    printf("%d, ", ptr[i]);  
}
```

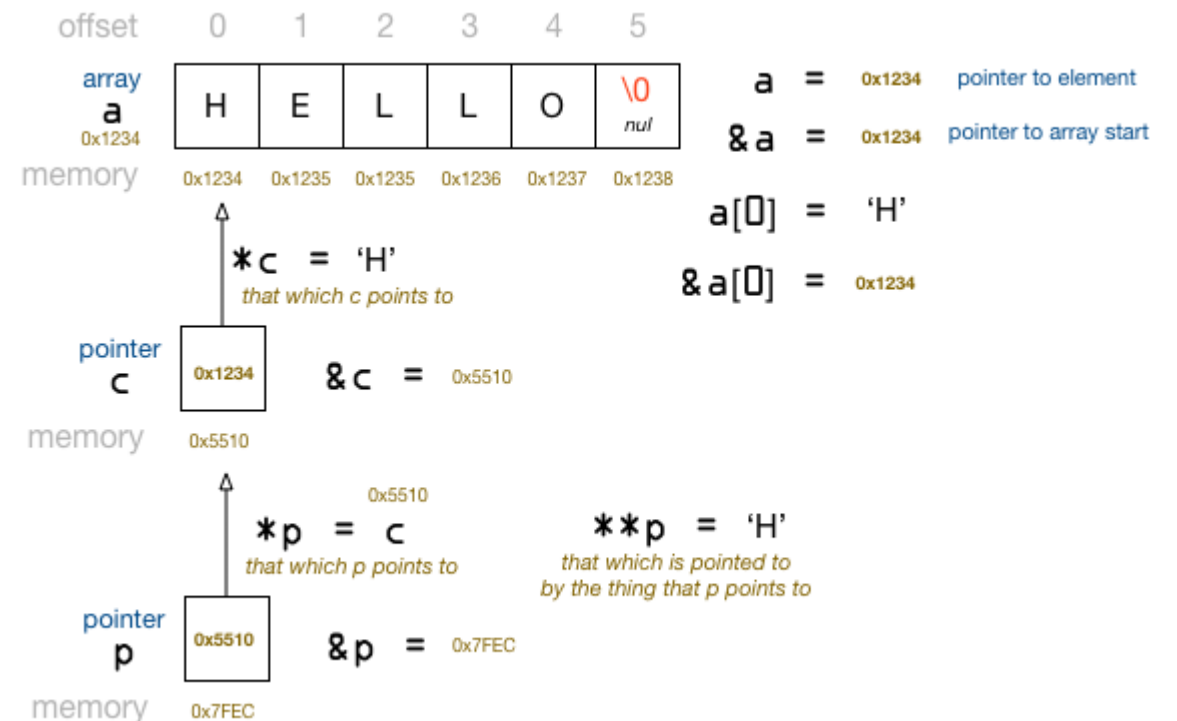


# Kaj pa string?

- Ni ni :)
- Imamo pa `char*` (juhu)
- Kaj dela `**str`

```
int main() {
    char* c = "Hello";
    char* c2 = malloc(6*sizeof(char));

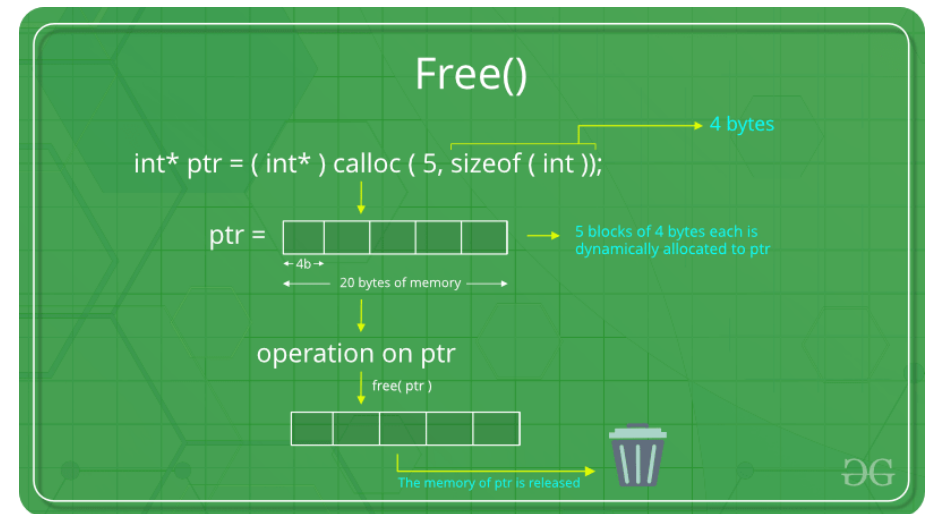
    char* p = c;
    printf("%c", *p);
}
```



# Kam z vsem pomnilnikom?

- Free!
- Ni garbage collectiona!
  - Če bi bil, ne bi govorili o pointerjih :)

```
ptr = (int*) malloc(100 * sizeof(int));  
free(ptr);
```



# printf

- Izpis
- Podobno kot v Javi

```
int main()
{
    printf ( "\n%f %f %f", 5.0, 13.5, 133.9 );
    printf ( "\n%f %f %f", 305.0, 1200.9,
3005.3 );
}
```

```
5.000000 13.500000 133.900000
305.000000 1200.900000 3005.300000
```

Data type		Format specifier
Integer	short signed	%d or %i
	short unsigned	%u
	long signed	%ld
	long unsigned	%lu
	unsigned hexadecimal	%x
	unsigned octal	%o
Real	float	%f
	double	%lf
Character	signed character	%c
	unsigned character	%c
String		%s

<https://c-programmingbooks.blogspot.com/2011/11/format-specifications-in-c-programming.html>

# printf

- Specifier length
- Zaokroževanje!

```
int main()
{
    printf ( "\n%10.1f %10.1f %10.1f", 5.0, 13.5,
133.9876 );
    printf ( "\n%10.1f %10.1f %10.1f", 305.0, 1200.95,
3005.96 );
}
```

```
    5.0      13.5      134.0
 305.0    1201.0    3006.0
```

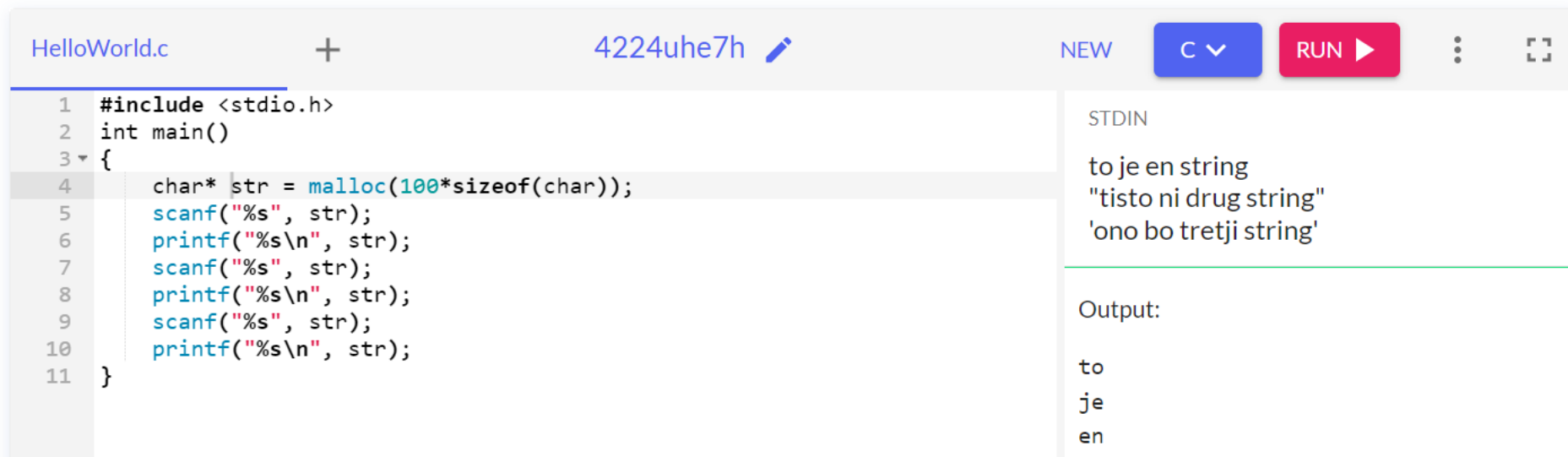
Specifier	Description
dd	Digits specifying field width
.	Decimal point separating field width from precision (precision stands for the number of places after the decimal point)
dd	Digits specifying precision
-	Minus sign for left justifying the output in the specified field width

<https://c-programmingbooks.blogspot.com/2011/11/format-specifications-in-c-programming.html>



# scanf

- Enaki specifier-ji
- Problem niza (string\*)
- Preverjanje dolžine (malloc)
- Kaj je alternativa?



```
1 #include <stdio.h>
2 int main()
3 {
4     char* str = malloc(100*sizeof(char));
5     scanf("%s", str);
6     printf("%s\n", str);
7     scanf("%s", str);
8     printf("%s\n", str);
9     scanf("%s", str);
10    printf("%s\n", str);
11 }
```

STDIN

to je en string  
"tisto ni drug string"  
'ono bo tretji string'

Output:

to  
je  
en

# gets

## Stack Canary



- “alternativa” [beware, it ain’t]
- Problemi (kaj se zgodi, ko je alokacija premajhna?)
- Kaj je stack smashing?

```
HelloWorld.c  +  4224uhe7h  NEW  C  RUN
```

```
1  #include <stdio.h>
2  int main()
3  {
4      char str[10];
5      gets(str);
6      printf("%s\n", str);
7      gets(str);
8      printf("%s\n", str);
9      gets(str);
10     printf("%s\n", str);
11     fflush(stdout);
12 }
13
14
```

STDIN

to je en string  
"tisto ni drug string dolg več kot deset znakov in zna bit problem"

Output:

to je en string  
"tisto ni drug string dolg več kot deset znakov in zna bit problem"  
'ono bo tretji string'

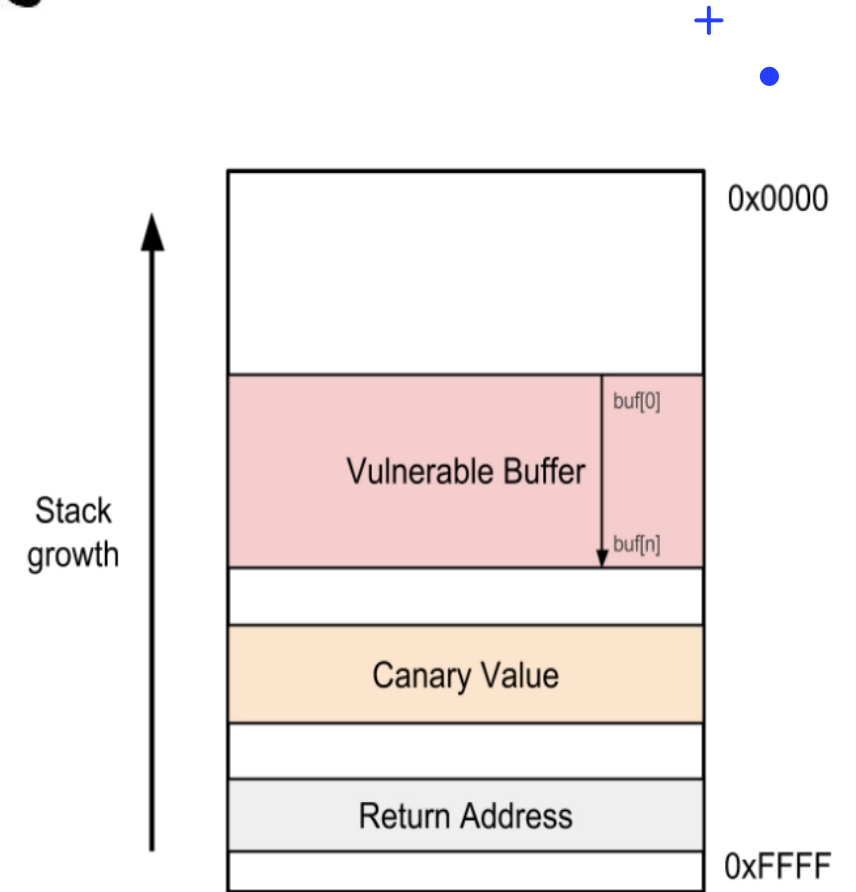
\*\*\* stack smashing detected \*\*\*: terminirano

# Stack canary

Stack Canary

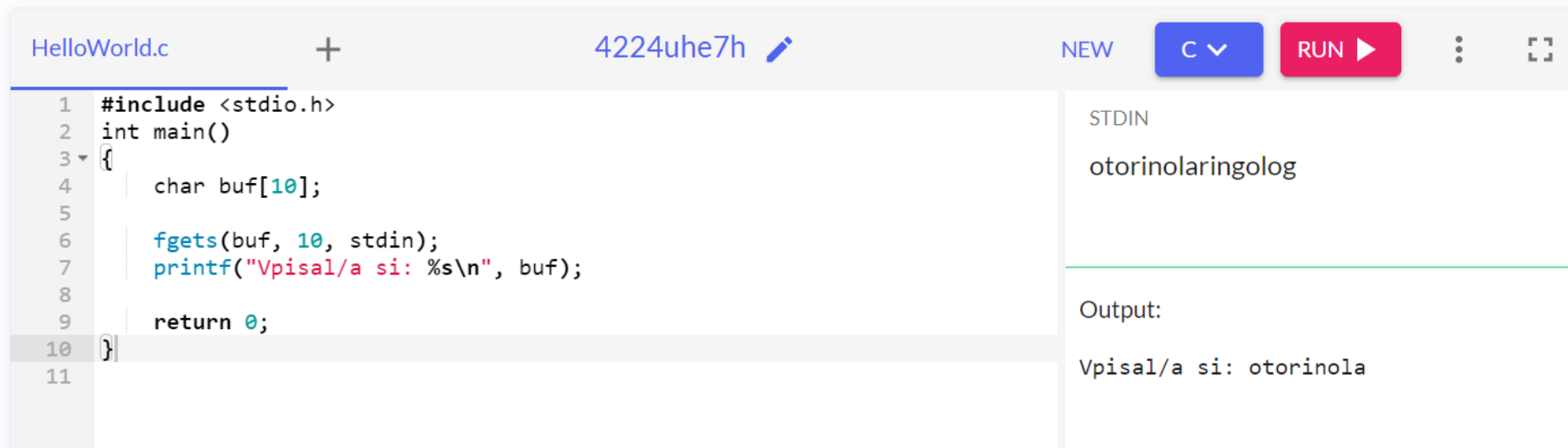


- Kanarčki v rudnikih (sklad)
- Prepišemo vrednosti, ki jih ne bi smeli
- Proces se zaključi, saj ne more nadaljevati
- Je gets najboljši način?



# fgets

- Boljša alternativa
- Problemi (kaj se zgodi, ko je alokacija premajhna?)
- Koliko črk smo shranili v buf?



The screenshot shows a code editor window titled 'HelloWorld.c' with a file icon and a user identifier '4224uhe7h'. The code is as follows:

```
1 #include <stdio.h>
2 int main()
3 {
4     char buf[10];
5
6     fgets(buf, 10, stdin);
7     printf("Vpisal/a si: %s\n", buf);
8
9     return 0;
10 }
11
```

On the right side of the editor, there are buttons for 'NEW', 'C' (with a dropdown arrow), and 'RUN' (with a play icon). Below these buttons, the 'STDIN' input is shown as 'otorinolaringolog'. The 'Output' section displays 'Vpisal/a si: otorinola'.

# Dodatna gradiva

## Programiranje v C

- Izbirni predmet (Dobravec)
- DIY
- Čim več prakse!

## Razlogi

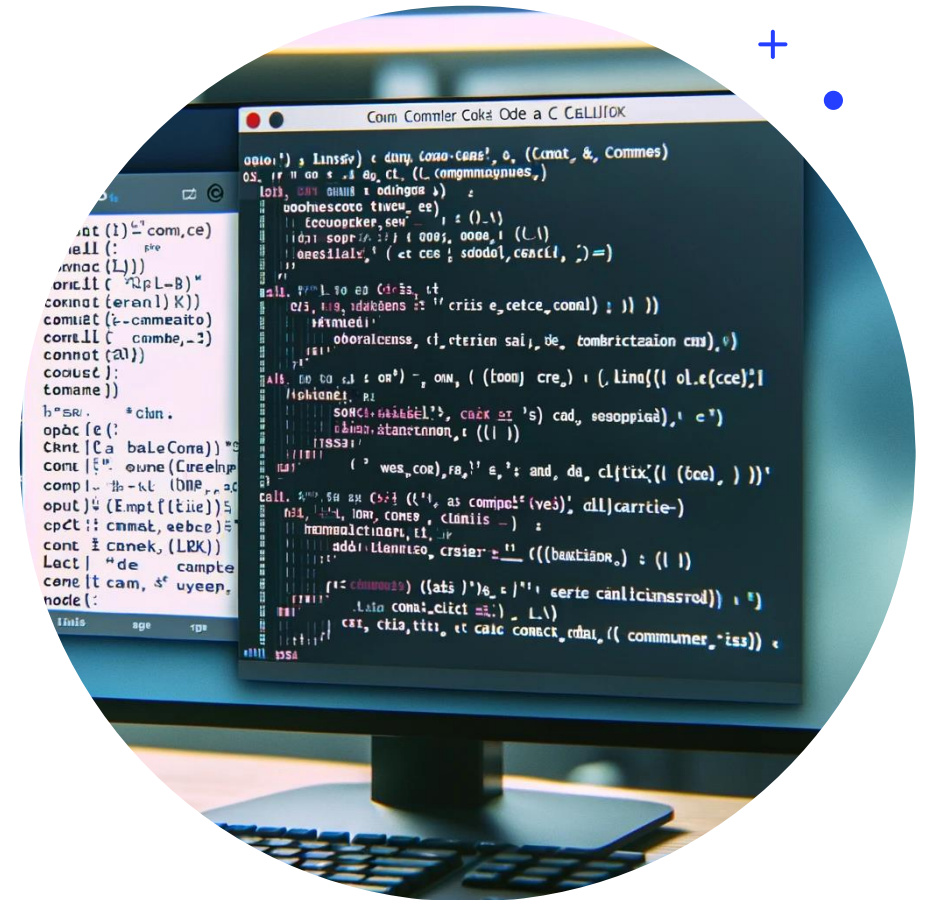
- Decompiling
- Disassembly
- Ni vezano zgolj na x86 platformo
  - iOS/Android

## Uporabna vrednost

- V CyberSec področju = 100% nujno
- Pri predmetu pri cca 20% vajah

# Hands-on

- Online c compiler ->  
<https://onecompiler.com/c/>
- Bash
  - Virtualka (katerakoli)\*
  - Windows subsystem ->  
<https://learn.microsoft.com/en-us/windows/wsl/about>





# BASH

World's best shell\*





# IZKUŠNJE?

Dvignite roke





# Zakaj Bash?

- Lahko tudi sh (bash ima mnogo več funkcionalnosti)
- Povsod na voljo (od ~80ih dalje)
- Vsi večji zaledni sistemi so Linux/Unix
- Poznate ga kot “ukazno lupino” (command prompt)



```
#!/bin/bash  
echo "Hello World"
```

# Ukazna lupina

- Včasih primarni vmesnik za delo
- Navidezno neskončni trak (gor/dol, ctrl+r)
- Znani ukazi
  - cat, echo, ls, grep, pwd, df, cd itd.
  - Kup stikal (ki niso nujno konsistentna)
    - -l -a -s (včasih tudi kot key/value pari)
- Skripte
  - Spisek ukazov za neinteraktivno delo

```
#!/bin/bash  
echo "Hello World"
```

# Enostavnost

- Občasno (pri delu z datotekami) neskončno uporabnejši od visokonivojskih jezikov
  - `cat /etc/passwd | cut -d: -f7 | sort | uniq -c | sort -gr`
- Dostop do celotnega datotečnega sistema
  - Tudi do psevdodatotečnih sistemov (npr. /proc)

```
#!/bin/bash  
echo "Hello World"  
exit 0
```

# Kompleksnost



- Veliko “legacy” kode
  - Vsakič ko se v Bashu vprašate “zakaj”, je odgovor “legacy”
- Čudne stvari
  - “\$a” vs. ‘\$a’
  - Sintaksa včasih ni sintaksa
    - a=10 vs. a = 10
    - Pogoji “[” in “[[”

```
#!/bin/bash
read -p "Vpiši ime: " name
echo "Pozdravljen/a, $name!"
echo ""
```

# Dodatna gradiva

## Operacijski sistemi

- VSŠ predmet
- UNI predmet
  - Vaje – prosojnice
- Online (neskončno veliko)
- Manual

## Razlogi

- Povsod bo na voljo bash
  - Celo na Windowsih:)
- Pomembno je razumeti delovanje OS
- Vse nove stvari vsebujejo ta "legacy"

## Uporabna vrednost

- V CyberSec področju = 100% nujno
- Za osebno uporabo zanesljivo najboljša stvar
  - Ocena pred prihodom ChatGPT ;)



# ASSEMBLY

It's awesome ... in a way :)



# Zbirnik (assembly)

## Osnove

- Na dnu sklada vseh ogrodi, jezikov, knjižnic
- Naši dedi so včasih luknjali kartice ... :)
- Še vedno uporaben za razumevanje, cybersec, izkoriščanje lukenj

## Ali ga rabimo (razumeti)?

- S bistvu ja/ne.
- Lahko ga pa razumemo do nam uporabne mere



# Enostaven primer

```
#include <stdio.h>
#include <unistd.h>

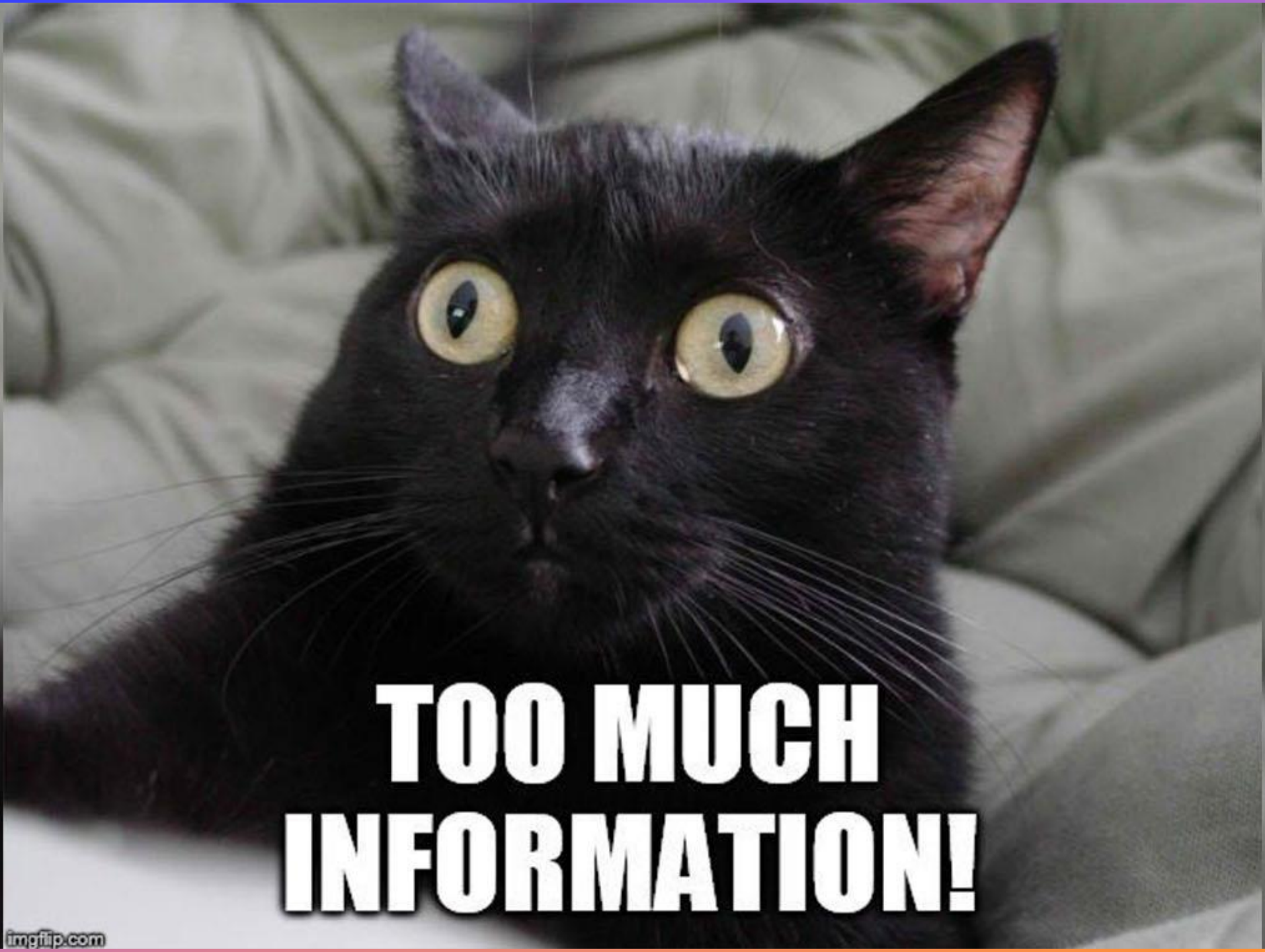
int main()
{
    write(1, "Hello World!\n", 13);
    return 0;
}
```

Dump of assembler code for function main:

```
=> 0x000055555555140 <+0>:    push    rbp
0x000055555555141 <+1>:    mov     rbp, rsp
0x000055555555144 <+4>:    sub     rsp, 0x10
0x000055555555148 <+8>:    mov     DWORD PTR [rbp-0x4], 0x0
0x00005555555514f <+15>:   mov     edi, 0x1
0x000055555555154 <+20>:   lea     rsi, [rip+0xea9]          #
0x555555556004
0x00005555555515b <+27>:   mov     edx, 0xd
0x000055555555160 <+32>:   call    0x55555555030 <write@plt>
0x000055555555165 <+37>:   xor     eax, eax
0x000055555555167 <+39>:   add     rsp, 0x10
0x00005555555516b <+43>:   pop     rbp
0x00005555555516c <+44>:   ret
```

End of assembler dump.





**TOO MUCH  
INFORMATION!**

# Kam naprej?

## Plan predmeta

- Sproti se bomo učili
- Veliko stvari lahko naredite v okviru termina vaj



## Osebni plan

- It's up to you
- Vedno podpiramo ideje izven okvirja
  - pridite na predavanja/vaje in jih delite z mano/nami
- Če še niste poskusili nečesa, je zdaj zadnji čas!

+



o



•



# HVALA

Vaje

- ponovitev C
- Spoznavanje okolja