**programiranje i programsko inženjerstvo**

**20.09.2019.**

**Zadatak 1.**

Što će se izvršavanjem sljedećeg programa ispisati na standardni izlaz (stdout)?

#include <stdio.h>

char \*fun(char \*s)  
{  
 static int i = 0;  
 int j = 3;  
 \*(s + i++) = \*(s + ++j);  
 return s;  
}

int main(void)  
{  
 char s[10] = "Vertical";  
 printf("%s", fun(s) + 1);  
 printf("%s", fun(s + 1));  
 return 0;  
}

|  |
| --- |
| ertical eatical |
| iertical ectical |
| ertical ectical |
| ertical iirtical |
| ertical crtical |

**Objašnjenje**

Kao argument u prvom pozivu zadane funkcije predaje se znakovni niz "Vertical". Pošto se argument daje kao pokazivač, promjene napravljene nad znakovnim nizom ostaju i nakon što se napusti funkcija.

Prvi poziv funkcije:

char \*fun(char \*s) // kao argument se prima "Vertical"  
{  
 static int i = 0; // vrijedi i = 0  
 int j = 3; // vrijedi j = 3  
 \*(s + i++) = \*(s + ++j); // na lokaciju i se postavlja vrijednost  
 return s; // s lokacije (j + 1) nakon čega se i  
} // povećava

Funkcija vraća pokazivač na znakovni niz u kojem je pohranjeno "tertical".

U prvom ispisu se ispisuje znakovni niz od druge pozicije jer je pokazivač na niz povećan za jedan.

printf("%s", fun(s) + 1); // ispis "ertical"

Kao argument u drugom pozivu zadane funkcije predaje se znakovni niz koji počinje od druge pozicije, odnosno, "ertical".

Drugi poziv funkcije:

char \*fun(char \*s) // kao argument se prima "ertical"  
{  
 static int i = 0; // vrijedi i = 1 (zbog prvog poziva)  
 int j = 3; // vrijedi j = 3  
 \*(s + i++) = \*(s + ++j); // na lokaciju i se postavlja vrijednost  
 return s; // s lokacije (j + 1) nakon čega se i  
} // povećava

Funkcija vraća pokazivač na znakovni niz "ectical" koji se potom ispisuje.

**20.09.2019.**

**Zadatak 2.**

Uz pretpostavku da se za pohranu podataka tipa int koristi 32 bita, koja vrijednost je pohranjena u varijablu j po završetku izvršavanja sljedećeg odsječka programa?

int i, j = 0;  
for (i = 1; i & 63; i = i << 2)  
{  
 j = j + i;  
}

|  |
| --- |
| 0 |
| 21 |
| 64 |
| 1431655765 |
| ne može se odrediti zbog beskonačne petlje |

**Objašnjenje**

Petlju u zadatku vrtimo sve dokle funkcija i nad binarnim brojem u varijabli i te binarnim zapisom broja 63 daje rezultat različit od nule.

Broj 63 zapisan u binarnom obliku (u 32 bita):

00000000000000000000000000111111

Prvi prolaz kroz petlju:

00000000000000000000000000111111

00000000000000000000000000000001

U prvom prolazu kroz petlju vrijedi i = 1 te j = 1. Nakon završetka prolaza, vrijednost u varijabli se pomiče za dva mjesta ulijevo.

Drugi prolaz kroz petlju:

00000000000000000000000000111111

00000000000000000000000000000100

U drugom prolazu kroz petlju vrijedi i = 4 te j = 1 + 4 = 5.

Treći prolaz kroz petlju:

00000000000000000000000000111111

00000000000000000000000000010000

U trećem prolazu kroz petlju vrijedi i = 16 te j = 5 + 16 = 21.

Nakon trećeg prolaza u varijablu i zapisana je vrijednost 1000000 (izostavljene nule na početku) te uvjet for petlje više nije zadovoljen i program izlazi iz petlje. Vrijednost varijable j na kraju iznosi 21.

**21.09.2018.**

**Zadatak 1.**

Ako se neka varijabla upiše u formatiranu datoteku, upisani broj bajtova u odnosu na veličinu te varijable u memoriji izražene u bajtovima bit će?

|  |
| --- |
| uvijek manji |
| uvijek veći |
| uvijek jednak |
| može biti manji, jednak ili veći |
| broj bajtova u datoteci i veličina varijable u memoriji ne mogu se uspoređivati |

**Objašnjenje**

**21.09.2018.**

**Zadatak 2.**

Uz pretpostavku da je varijabla b smještena u memoriji neposredno iza varijable a, slijedeći odsječak ispisat će?

int b = 1, a = 2;  
int \*p;  
p = &a;  
\*p = 7;  
(\*p)++;  
\*(++p) = 8;  
printf("%d %d\n", a, b);

|  |
| --- |
| 2 1 |
| 8 8 |
| 7 1 |
| 8 1 |
| 2 8 |

**Objašnjenje**

Isječak programa stvara pokazivač na varijablu a. Prvo se mijenja vrijednost pohranjena na lokaciji na koju pokazivač pohranjuje, odnosno, mijenja se vrijednost varijable a, zatim se mijenja vrijednost varijable na lokaciji iza, odnosno, mijenja se vrijednost varijable b.

p = &a; // pokazivač na varijablu a  
\*p = 7; // vrijednost varijable a mijenja se u 7  
(\*p)++; // vrijednost varijable a povećava se za 1  
\*(++p) = 8; // vrijednost varijable b mijenja se u 8

Konačno, u obje varijable pohranjena je vrijednost 8 stoga se ispisuje 8 8.

**22.09.2017.**

**Zadatak 1.**

Što će se izvršavanjem sljedećeg programa ispisati na standardni izlaz (stdout)?

#include <stdio.h>

char fun1(char s)  
{  
 return s + 2;  
}

char \*fun2(char \*s)  
{  
 return s + 2;  
}

int main(void)  
{  
 char numbers[10] = {'5', '4', '3', '2', '1'};  
 printf("%d", fun1(\*numbers));  
 printf("%s", fun2(numbers));  
 return 0;  
}

|  |
| --- |
| 55321 |
| 73 |
| 354 |
| 7321 |
| 54323321 |

**Objašnjenja**

Prvo se poziva funkcija fun1 koja kao argument prima jedan znak. Funkciji je predan \*s što predstavlja prvi element polja numbers, odnosno element numbers[0]. Funkcija vraća vrijednost znaka uvećanu za dva, odnosno znak '7', no kako funkcija printf ispisuje broj, umjesto znaka '7' ispisuje se njegova ASCII vrijednost 55.

Potom se poziva funkcija fun2 koja kao argument prima znakovni niz, a vraća pokazivač na isti znakovni niz, no od drugog znaka umjesto prvog. Naime, zbog s+2 u funkciji dolazi do pomicanja pokazivača na memorijsku lokaciju koja je za dva mjesta udaljenija. Funkciji je kao argument predano cijelo polje znakova, odnosno znakovni niz, no vraća se samo pokazivač na elemente na mjestima 2 i više, stoga je ispis jednak 3 2 1.

**22.09.2017.**

**Zadatak 2.**

Uz pretpostavku da se za pohranu podatka tipa int koristi 32 bita, koja vrijednost je pohranjena u varijablu j po završetku izvršavanja sljedećeg odsječka programa?

unsigned int i = 77, j = 0;  
while (i & 5)  
{  
 i = i >> 1;  
 j = j + (i & 7);  
}

|  |
| --- |
| 5 |
| 14 |
| 2 |
| 16 |
| ne može se odrediti zbog beskonačne petlje |

**Objašnjenje**

Vrijednost varijable i zapisane u binarnom zapisu glasi:

00000000000000000000000001001101

Petlja će se pokretati sve dokle funkcija logičko i binarnog zapisa varijable i te binarnog zapisa broja 5 bude davala rezultat različit od 0.

Prvi prolaz petlje:

i = 00000000000000000000000001001101  
i = i >> 1 = 00000000000000000000000000100110  
j = 6

Drugi prolaz petlje:

i = 00000000000000000000000000100110  
i = i >> 1 = 00000000000000000000000000010011  
j = 6 + 3 = 9

Treći prolaz petlje:

i = 00000000000000000000000000010011  
i = i >> 1 = 00000000000000000000000000001001  
j = 9 + 1 = 10

Četvrti prolaz petlje:

i = 00000000000000000000000000001001  
i = i >> 1 = 00000000000000000000000000000100  
j = 10 + 4 = 14

Peti prolaz petlje:

i = 00000000000000000000000000000100  
i = i >> 1 = 00000000000000000000000000000010  
j = 14 + 2 = 16

Nakon petog prolaza petlje, vrijednost varijable i više ne zadovoljava uvjet za ponovno pokretanje petlje te petlja prekida s vrijednosti j = 16.

**19.07.2017.**

**Zadatak 1.**

Što će se izvršavanjem sljedećeg programa ispisati na standardni izlaz (stdout)?

#include <stdio.h>

int i = 2;

void fun(int j)  
{  
 static int i = 5;  
 printf("%2d%2d", i++, j);  
}

int main(void)  
{  
 fun(i++);  
 fun(++i);  
 return 0;  
}

|  |
| --- |
| 5 2 5 4 |
| 5 2 6 4 |
| 2 2 2 4 |
| 2 2 3 3 |
| 5 3 6 4 |

**Objašnjenje**

Funkcija fun unutar sebe koristi statičku varijablu i, dok glavni program koristi globalnu varijablu i. To znači da će svaki poziv funkcije fun kao argument primati vrijednost iz globalne i varijable.

void fun(int j)  
{  
 static int i = 5;  
 printf("%2d%2d", i++, j); // koristi se i varijabla funkcije  
}

int main(void)  
{  
 fun(i++); // koristi se globalna i varijabla  
 fun(++i); // koristi se globalna i varijabla  
 return 0;  
}

Prvi poziv funkcije kao argument šalje globalnu varijablu s pohranjenom vrijednosti 2 koja se nakon poziva funkcije povećava na 3. Ispis iz prvog poziva funkcije glasi 5 2 pri čemu se korištena vrijednost statičke varijable nakon ispisa poveća za 1.

Drugi poziv funkcije kao argument šalje globalnu varijablu koja se prije slanja poveća za jedan. Uzimajući u obzir i povećanje iz prethodnog poziva, vrijednost varijable argumenta iznosi 4. Ispis iz drugog poziva funkcije glasi 6 4.

Konačno, cjelokupni ispis programskog odsječka glasi 5 2 6 4.

**19.07.2017.**

**Zadatak 2.**

Koja je vrijednost pohranjena u varijablu j po završetku izvršavanja sljedećeg odsječka programa?

int j;  
int x[3][4] = {{1, 3, 5}, {7, 9, 11}, {13, 15, 17}};  
int \*p1 = &x[1][1];  
int \*p2 = p1 + 3;  
j = \*p1 + \*(p1 + 2) + \*p2 + \*(p2 + 2);

|  |
| --- |
| 11 |
| 16 |
| 20 |
| 39 |
| 52 |

**Objašnjenje**

U odsječku programa stvara se dvodimenzionalno polje s tri retka i četiri stupca pri čemu su vrijednosti zadane samo za prva tri stupca.

Prvi pokazivač pokazuje na element u prvom retku i prvom stupcu. Drugi pokazivač pokazuje na memorijsku lokaciju koja je za tri udaljena od lokacije na koju pokazuje prvi pokazivač, odnosno, na element u drugom retku i nultom stupcu.

Konačno, vrijednost varijable j računa se kao zbroj vrijednosti na lokaciji na koju pokazuje prvi pokazivač, vrijednosti na lokaciji koja je udaljena za dva od prve, vrijednosti na lokaciji na koju pokazuje drugi pokazivač i vrijednosti na lokaciji koja je udaljena za dva od druge.

j = \*p1 + \*(p1 + 2) + \*p2 + \*(p2 + 2) = 9 + 0 + 13 + 17 = 39

**23.09.2016.**

**Zadatak 1.**

Što će se izvršavanjem sljedećeg programa ispisati na standardni izlaz (stdout)?

#include <stdio.h>

void fun(char \*s)  
{  
 \*s = \*s + 1;  
 printf("%c", \*s);  
 s = s + 1;  
}

int main(void)  
{  
 char name[] = "Arnie";  
 char \*s = name;  
 fun(s);  
 fun(s + 1);  
 fun(name);  
 return 0;  
}

|  |
| --- |
| BsC |
| AsC |
| ArA |
| Bop |
| BoC |

**Objašnjenje**

Funkcija kao argument prima znakovni niz ili pokazivač na polje znakova te povećava vrijednost na lokaciji na koju pokazivač pokazuje za 1. Linija s = s + 1 ne mijenja pokazivač niti vrijednost na lokaciji na kojoj pokazuje.

int main(void)  
{  
 char name[] = "Arnie";  
 char \*s = name;  
 fun(s); // A se mijenja s B  
 fun(s + 1); // r se mijenja sa s  
 fun(name); // B se mijenja s C  
 return 0;  
}

**23.09.2016.**

**Zadatak 2.**

Koja bi od ponuđenih naredbi, ako bi se umetnula na mjesto označeno s tri točke, spriječila uspješno prevođenje sljedećeg programa?

#include <stdio.h>  
#define POSITION 3

int main(void)  
{  
 int numbers[5] = {1, 2, 3, 4, 5};  
 ...  
 printf("%d", numbers[3]);  
 return 0;  
}

|  |
| --- |
| \*numbers = ('H' + POSITION); |
| \*numbers = \*(numbers + POSITION); |
| \*(numbers + POSITION) = numbers[1] |
| numbers[POSITION / 10] = POSITION/2; |
| numbers = numbers + POSITION; |

**Objašnjenje**

Nije definirano zbrajanje između polja i broja.

**20.07.2016.**

**Zadatak 1.**

Što će se izvršavanjem sljedećeg programa ispisati na standardni izlaz (stdout)?

#include <stdio.h>

int fun(int \*x, int i)  
{  
 int r;  
 r = \*x + i / 2 + \*(x + i / 2);  
 return r;  
}

int main(void)  
{  
 int x[] = {8, 7, 6, 5, 4, 3, 2, 1};  
 printf("%d", fun(x, 7));  
 return 0;  
}

|  |
| --- |
| 7 |
| 10 |
| 12 |
| 16 |
| 22 |

**Objašnjenje**

Funkcija kao argument prima pokazivač na polje cijelih brojeva i cijeli broj, a vraća zbroj vrijednosti prvog elementa polja, cjelobrojnog kvocijenta predanog cijelog broja i broja 2 te vrijednosti elementa koji je za cjelobrojni kvocijent predanog cijelog broja i broja 2 udaljen od lokacije na koju pokazivač pokazuje.

r = \*x + i / 2 + \*(x + i / 2) = 8 + 3 + x[3] = 11 + 5 = 16

**20.07.2016.**

**Zadatak 2.**

Logički izraz u sljedećoj naredbi

if (!(x >= 3 || y < 5 && z != 15))  
{  
 printf("C program");  
}

može se zamijeniti ekvivalentnim logičkim izrazom:

|  |
| --- |
| ! x >= 3 || ! y < 5 && ! z ! = 15 |
| !( x >= 3) || ! (y < 5) && !( z ! = 15) |
| (x < 3 || y >= 5) && z == 15 |
| x < 3 && (y >= 5 || z == 15) |
| (x < 3 && y >= 5) || (z == 15) |

**Objašnjenje**

DeMorganovo pravilo.