Argumenti komandne linije

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je dobijena izvršna datoteka pokrenuta komandom `./program a b`:

```
#include <stdio.h>


int main(int num_args, char *args[]) {
   puts(args[0]);


   return 0;
}
```

- `./program`
- `./program a`
- `./program a b`
- `a`
- `a b`
- `b`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je dobijena izvršna datoteka pokrenuta komandom `./program a`:

```
#include <stdio.h>


int main(int num_args, char *args[]) {
    puts(args[1]);


    return 0;
}
```

- `a`
- `./program`
- `./program a`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je dobijena izvršna datoteka pokrenuta komandom `./program a`:

```
#include <stdio.h>


int main(int num_args, char *args[]) {
    printf("%d, %s", num_args, args[1]);


    return 0;
}
```

- `2, a`

- `1, a`
- `1, ./program`
- `2, ./program`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je dobijena izvršna datoteka pokrenuta komandom `./program a`:

```
#include <stdio.h>


int main(int num_args, char *args[]) {
    printf("%d, %s", num_args, args[0]);


    return 0;
}
```

- `2, ./program`
- `1, a`
- `2, a`
- `1, ./program`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je dobijena izvršna datoteka pokrenuta komandom `./program a b c d`:

```
#include <stdio.h>


int main(int num_args, char *args[]) {
    char **p = args;


    puts(*args);


    return 0;
}
```

- `./program`
- `./program a b c d`
- `a`
- `b`
- `c`
- `d`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je dobijena izvršna datoteka pokrenuta komandom `./program a b c d`:

```
#include <stdio.h>


int main(int num_args, char *args[]) {
```

```
    char **p = args;

    puts(*(args+num_args-1));

    return 0;
}
```

- d
- ./program
- ./program a b c d
- a
- b
- c
- d c b a ./program

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je dobijena izvršna datoteka pokrenuta komandom `./program a b`:

```
#include <stdio.h>

int main(int num_args, char **args) {
    char **p = args;

    while(num_args) {
        printf("%s ", *args);

        num_args--;
    }

    return 0;
}
```

- ./program ./program ./program
- ./program
- ./program ./program
- ./program a b
- a b
- b a
- b a ./program

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je dobijena izvršna datoteka pokrenuta komandom `./program a b`:

```
#include <stdio.h>

int main(int num_args, char **args) {
```

```
    char **p = args;

    while(num_args) {
        num_args--;

        printf("%s ", *(args+num_args));
    }

    return 0;
}
```

- b a ./program
- ./program
- ./program ./program
- ./program ./program ./program
- ./program a b
- a b
- b a

Bitwise operatori

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    char x = 72; // 72 == 0100 1000

    while(x = 0) {
        puts("PJISP");

        x>>=1;
    }

    puts("PJISP");

    return 0;
}
```

- 1 put
- 3 puta

- 4 puta
- 5 puta
- 6 puta
- 7 puta
- 8 puta
- 16 puta
- beskonačno mnogo puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    char x = 72; // 72 == 0100 1000


    while(x != 0) {
        puts("PJISP");


        x>>=1;
    }


    return 0;
}
```

- 7 puta
- 1 put
- 3 puta
- 4 puta
- 5 puta
- 6 puta
- 8 puta
- 16 puta
- beskonačno mnogo puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    char x = 72; // 72 == 0100 1000


    while(x != 0) {
        puts("PJISP");
```

```
        x>>=2;
    }

    return 0;
}
```

- 4 puta
- 1 put
- 3 puta
- 5 puta
- 6 puta
- 7 puta
- 8 puta
- 16 puta
- beskonačno mnogo puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    char x = 72; // 72 == 0100 1000

    while(x != 0) {
        puts("PJISP");

        x<<=1;
    }

    return 0;
}
```

- 5 puta
- 1 put
- 3 puta
- 4 puta
- 6 puta
- 7 puta
- 8 puta
- 16 puta
- beskonačno mnogo puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

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

```
int main() {
    char x = 72; // 72 == 0100 1000

    while(x != 0) {
        puts("PJISP");

        x<<=2;
    }

    return 0;
}
```

- 3 puta
- 1 put
- 4 puta
- 5 puta
- 6 puta
- 7 puta
- 8 puta
- 16 puta
- beskonačno mnogo puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    char x = 0x8;

    while(x != 0) {
        puts("PJISP");

        x<<=1;
    }

    return 0;
}
```

- 5 puta
- 1 puta
- 3 puta
- 8 puta
- 13 puta
- 16 puta

- beskonačno mnogo puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    short x = 0x8;

    while(x != 0) {
        puts("PJISP");

        x<<=1;
    }

    return 0;
}
```

- 13 puta
- 1 puta
- 3 puta
- 5 puta
- 8 puta
- 16 puta
- beskonačno mnogo puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    char x = 72; // 72 == 0100 1000

    while(x != 0) {
        puts("PJISP");

        x<<2;
    }

    return 0;
}
```

- beskonačno mnogo puta
- 1 put

- 3 puta
- 4 puta
- 5 puta
- 6 puta
- 7 puta
- 8 puta
- 16 puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    char x = 72; // 72 == 0100 1000


    while(x != 0) {
        puts("PJISP");


        x>>1;
    }


    return 0;
}
```

- beskonačno mnogo puta
- 1 put
- 3 puta
- 4 puta
- 5 puta
- 6 puta
- 7 puta
- 8 puta
- 16 puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    char x = 72; // 72 == 0100 1000


    while(x = 0) {
        puts("PJISP");
```

```
        x>>1;
    }

    puts("PJISP");

    return 0;
}
```

- 1 put
- 3 puta
- 4 puta
- 5 puta
- 6 puta
- 7 puta
- 8 puta
- 16 puta
- beskonačno mnogo puta

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    unsigned x = 72; // 72 == 0100 1000
    char i;

    i^=i;
    while(x != 0) {
        i++;
        x<<=1;
    }

    printf("%i", i);

    return 0;
}
```

- 29
- 1
- 3
- 4
- 6
- 7
- 8
- 13
- 30

- program će prilikom izvršavanja ući u beskonačnu petlju

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 1;
    int b = 2;


    if ((a ^ a))
        printf("Veci je b");
    else
        printf("Veci je a");


    return 0;
}
```

- Veci je a
- Veci je b
- ništa neće biti ispisano
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 1;
    int b = 2;


    if (!(a ^ a))
        printf("Veci je b");
    else
        printf("Veci je a");


    return 0;
}
```

- Veci je b
- Veci je a
- ništa neće biti ispisano
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 1;
    int b = 2;


    if (!!(a ^ a))
        printf("Veci je b");
    else
        printf("Veci je a");


    return 0;
}
```

- Veci je a
- Veci je b
- ništa neće biti ispisano
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 1;
    int b = 2;


    if (!!!(a ^ a))
        printf("Veci je b");
    else
        printf("Veci je a");


    return 0;
}
```

- Veci je b
- Veci je a
- ništa neće biti ispisano
- prilikom kompajliranja dobijamo grešku/upozorenje

Kojim bitwise operatorom se postavlja vrednost željenog bita na 1?

- |

- `~`
- `!`
- `&`
- `&&`
- `||`
- `^`

Kog tipa treba biti promenljiva `abc` u sledećem delu koda:

```
xyz = fopen(abc, "w");
if(xyz == NULL) {
    exit(EXIT_FAILURE);
}
```

- string (tj. `char *`)
- `char`
- `FILE`
- `FILE *`
- `float`
- `int`
- `unsigned`
- režim za čitanje
- režim za pisanje
- režim za dodavanje

Kog tipa treba biti promenljiva `xyz` u sledećem delu koda:

```
xyz = fopen(abc, "w");
if(xyz == NULL) {
    exit(EXIT_FAILURE);
}
```

- `FILE *`
- `char`
- `FILE`
- `float`
- `int`
- string (tj. `char *`)
- `unsigned`
- režim za čitanje
- režim za pisanje
- režim za dodavanje

Šta će biti sadržaj tekstualne datoteke "pjisp.txt" nakon izvršavanja sledećeg koda, ukoliko je u datoteci prethodno bio zapisan tekst "123":

```
#include <stdio.h>


int main() {
    FILE *out = fopen("pjisp.txt", "a");
    fprintf(out, "456");


    fclose(out);


    return 0;
}
```

- 123456
- 123
- 456
- 456123
- datoteka će biti prazna

Šta će biti sadržaj tekstualne datoteke "pjisp.txt" nakon izvršavanja sledećeg koda, ukoliko je u datoteci prethodno bio zapisan tekst "123":

```
#include <stdio.h>


int main() {
    FILE *out = fopen("pjisp.txt", "w");
    fprintf(out, "456");


    fclose(out);


    return 0;
}
```

- 456
- 123
- 123456
- 456123
- datoteka će biti prazna

Šta će biti sadržaj tekstualne datoteke "pjisp.txt" nakon izvršavanja sledećeg koda, ukoliko je u datoteci prethodno bio zapisan tekst "123":

```
#include <stdio.h>


int main() {
    FILE *out = fopen("pjisp.txt", "r");
    fprintf(out, "456");
```

```
    fclose(out);

    return 0;
}
```

- 123
- 456
- 123456
- 456123
- datoteka će biti prazna

Šta će biti sadržaj tekstualne datoteke "pjisp.txt" nakon izvršavanja sledećeg koda, ukoliko je u datoteci prethodno bio zapisan tekst "123":

```
#include <stdio.h>

int main() {
    FILE *out = fopen("pjisp.txt", "w");
    fprintf(stdout, "456");

    fclose(out);

    return 0;
}
```

- datoteka će biti prazna
- 123
- 456
- 123456
- 456123

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je u datoteci "pjisp.txt" prethodno bio zapisan tekst "123456":

```
#include <stdio.h>

#define BUFF_SIZE 4096

int main() {
    char buff[BUFF_SIZE];
    FILE *in = fopen("pjisp.txt", "r");

    fscanf(in, "%c", buff);
    printf("%c", *buff);
```

```
    fclose(in);

    return 0;
}
```

- 1
- 12
- 123
- 1234
- 12345
- 123456
- ništa neće biti ispisano

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, ukoliko je u datoteci "pjisp.txt" prethodno bio zapisan tekst "123 456":

```
#include <stdio.h>

#define BUFF_SIZE 4096

int main() {
    char buff[BUFF_SIZE];
    FILE *in = fopen("pjisp.txt", "r");

    fscanf(in, "%s", buff);
    printf("%s", buff);

    fclose(in);

    return 0;
}
```

- 123
- 1
- 12
- 1234
- 12345
- 123456
- ništa neće biti ispisano

Kako se pravilno zauzima memorija za promenljivu `genesis_block`:

```
typedef struct block_st {
```

```
    int index;
    int hash;
    char num_tx;
    int timestamp;
    int nonce;

    struct block_st *parent_block;
} BLOCK;

BLOCK *genesis_block;
```

- `genesis_block = malloc(sizeof(BLOCK));`
- `genesis_block = malloc(sizeof(BLOCK*));`
- `genesis_block = malloc(sizeof(block_st));`
- `genesis_block = malloc(sizeof(block_st*));`
- `genesis_block = malloc(sizeof(int));`
- `free(genesis_block);`

Kako se pravilno oslobađa memorija zauzeta za promenljivu `genesis_block`:

```
typedef struct block_st {
    int index;
    int hash;
    char num_tx;
    int timestamp;
    int nonce;

    struct block_st *parent_block;
} BLOCK;

BLOCK *genesis_block;
```

- `free(genesis_block);`
- `genesis_block = malloc(sizeof(BLOCK));`
- `genesis_block = malloc(sizeof(BLOCK*));`
- `genesis_block = malloc(sizeof(block_st));`
- `genesis_block = malloc(sizeof(block_st*));`
- `genesis_block = malloc(sizeof(int));`

Koliko bajtova memorije će zauzeti sledeći (uspešni) poziv `malloc` funkcije:

```
struct grupa_st {
    int a;
    char b[30];
    char c;
```

```
};

struct grupa_st *g = malloc(sizeof(struct grupa_st));
```

- 35
- 4
- 30
- 31
- 34
- 36

Koliko bajtova memorije će zauzeti sledeći (uspešni) poziv `malloc` funkcije:

```
union grupa_st {
    int a;
    char b[30];
    char c;
};

union grupa_st *g = malloc(sizeof(union grupa_st));
```

- 30
- 4
- 31
- 34
- 35
- 36

Koju operaciju nad dvostruko spregnutom listom implementira sledeći deo koda:

```
typedef struct element_st {
    int broj;

    struct element_st *pret;
    struct element_st *sled;
} ELEMENT;

typedef struct lista_st {
    ELEMENT *prvi;
    ELEMENT *posl;
} LISTA;

// ... nebitan kod ...

LISTA *lst;
```

```
// ... nebitan kod ...

ELEMENT *tek;
for (tek=lst.posl; tek; tek=tek->pret);
```

- obilazak liste unazad
- inicijalizacija liste
- obilazak liste unapred
- unos novog elementa na početak liste
- unos novog elementa na kraj liste
- brisanje elementa iz liste
- brisanje liste

Koju operaciju nad dvostruko spregnutom listom implementira sledeći deo koda:

```
typedef struct element_st {
    int broj;

    struct element_st *pret;
    struct element_st *sled;
} ELEMENT;

typedef struct lista_st {
    ELEMENT *prvi;
    ELEMENT *posl;
} LISTA;

// ... nebitan kod ...

LISTA *lst;

// ... nebitan kod ...

ELEMENT *tek;
for (tek=lst.prvi; tek; tek=tek->sled);
```

- obilazak liste unapred
- inicijalizacija liste
- obilazak liste unazad
- unos novog elementa na početak liste
- unos novog elementa na kraj liste
- brisanje elementa iz liste
- brisanje liste

Koju operaciju nad jednostruko spregnutom listom karaktera implementira sledeći deo koda:

```
tek = glava;
while(tek != NULL) {
    printf("%c", tek->znak);

    tek = tek->sledeci;
}
```

- listanje liste
- inicijalizacija liste
- unos novog elementa na početak liste
- unos novog elementa na kraj liste
- brisanje elementa iz liste
- brisanje liste

Koju operaciju nad jednostruko spregnutom listom karaktera implementira sledeći deo koda:

```
tek = glava;
while(tek != NULL) {
    tek = tek->sledeci;
}
```

- prolazak do kraja liste
- inicijalizacija liste
- unos novog elementa na početak liste
- unos novog elementa na kraj liste
- brisanje elementa iz liste
- brisanje liste

Koju operaciju nad jednostruko spregnutom listom karaktera implementira sledeći deo koda:

```
tek = glava;
pret = glava;
while(tek != NULL && (tek->znak != c)) {
    pret = tek;
    tek = tek->sledeci;
}
```

- traženje elementa u listi
- inicijalizacija liste
- unos novog elementa na početak liste
- unos novog elementa na kraj liste
- brisanje elementa iz liste

- brisanje liste

Koju operaciju nad jednostruko spregnutom listom karaktera implementira sledeći deo koda:

```
while(glava != NULL) {
    tek = glava;
    glava = tek->sledeci;

    free(tek);
}
```

- brisanje liste
- inicijalizacija liste
- listanje liste
- unos novog elementa na početak liste
- unos novog elementa na kraj liste
- brisanje elementa iz liste

Koju operaciju nad jednostruko spregnutom listom karaktera implementira sledeći deo koda:

```
int f(BCVOR* node, int t) {
    if(node == NULL)
        return 0;

    if(t == 0)
        return 1;

    return f(node->left, t-1) + f(node->right, t-1);
}
```

- ovo nije operacija nad listom
- inicijalizacija liste
- listanje liste
- unos novog elementa na početak liste
- unos novog elementa na kraj liste
- brisanje elementa iz liste
- brisanje liste

Šta radi sledeća funkcija:

```
void f(struct node **front, struct node **rear, int value) {
    struct node *temp = malloc(sizeof(struct node));
    if(temp == NULL) {
```

```
        puts("Greska prilikom zauzimanja memorije!");
        exit(42);
    }

    temp->data = value;
    temp->link = NULL;

    if(*rear == NULL) {
        *rear = temp;
        *front = *rear;
    } else {
        (*rear)->link = temp;
        *rear = temp;
    }
}
```

- ubacuje novi element u red
- ubacuje novi element u binarni hip
- ubacuje novi element u stek
- uklanja element uz binarnog hipa
- uklanja element uz reda
- uklanja element iz steka

Koju operaciju nad binarnim stablom pristupa, uređenim tako da vrednost elemenata ne opada prilikom obilaska stabla sleva-udesno, implementira sledeći deo koda:

```
int f(BCVOR* node) {
    while (node->left != NULL) {
        node = node->left;
    }

    return node->data;
}
```

- pronalaženje najmanje vrednosti u stablu
- balansiranje stabla
- određivanje maksimalne dubine stabla
- proverava da li postoji putanja od korena do lista sa zadatom sumom elemenata
- računanje broja elemenata na zadatom nivou stabla
- računanje broja elemenata u stablu
- računanje sume elemenata stabla
- zamenu levog i desnog podstabla, za svaki element u stablu

Koju operaciju nad binarnim stablom pristupa, uređenim tako da vrednost elemenata ne opada prilikom obilaska stabla sleva-udesno, implementira sledeći deo koda:

```
int f(BCVOR* node) {
    while (node->right != NULL) {
        node = node->right;
    }

    return node->data;
}
```

- pronalaženje najveće vrednosti u stablu
- balansiranje stabla
- određivanje maksimalne dubine stabla
- proverava da li postoji putanja od korena do lista sa zadatom sumom elemenata
- računanje broja elemenata na zadatom nivou stabla
- računanje broja elemenata u stablu
- računanje sume elemenata stabla
- zamenu levog i desnog podstabla, za svaki element u stablu

Koju operaciju nad binarnim stablom pristupa, uređenim tako da vrednost elemenata ne opada prilikom obilaska stabla sleva-udesno, implementira sledeći deo koda:

```
int f(BCVOR* node) {
    if (node == NULL) {
        return 0;
    }

    int l = f(node->left);
    int r = f(node->right);

    return 1 + ((l > r) ? l : r);
}
```

- određivanje maksimalne dubine stabla
- balansiranje stabla
- pronalaženje najmanje vrednosti u stablu
- proverava da li postoji putanja od korena do lista sa zadatom sumom elemenata
- računanje broja elemenata na zadatom nivou stabla
- računanje broja elemenata u stablu
- računanje sume elemenata stabla
- zamenu levog i desnog podstabla, za svaki element u stablu

Koju operaciju nad binarnim stablom pristupa, uređenim tako da vrednost elemenata ne opada prilikom obilaska stabla sleva-udesno, implementira sledeći deo koda:

```
int f(BCVOR* node) {
    if (node == NULL) {
        return 0;
    }

    return 1 + f(node->left) + f(node->right);
}
```

- računanje broja elemenata u stablu
- balansiranje stabla
- određivanje maksimalne dubine stabla
- pronalaženje najmanje vrednosti u stablu
- proverava da li postoji putanja od korena do lista sa zadatom sumom elemenata
- računanje broja elemenata na zadatom nivou stabla
- računanje sume elemenata stabla
- zamenu levog i desnog podstabla, za svaki element u stablu

Koju operaciju nad binarnim stablom pristupa, uređenim tako da vrednost elemenata ne opada prilikom obilaska stabla sleva-udesno, implementira sledeći deo koda:

```
int f(BCVOR* node) {
    if (node == NULL) {
        return 0;
    }

    return node->data + f(node->left) + f(node->right);
}
```

- računanje sume elemenata stabla
- balansiranje stabla
- određivanje maksimalne dubine stabla
- pronalaženje najmanje vrednosti u stablu
- proverava da li postoji putanja od korena do lista sa zadatom sumom elemenata
- računanje broja elemenata na zadatom nivou stabla
- računanje broja elemenata u stablu
- zamenu levog i desnog podstabla, za svaki element u stablu

Koju operaciju nad binarnim stablom pristupa, uređenim tako da vrednost elemenata ne opada prilikom obilaska stabla sleva-udesno, implementira sledeći deo koda:

```
int f(BCVOR* node, int t) {
    if(node == NULL)
        return 0;
```

```
    if(t == 0)
        return 1;

    return f(node->left, t-1) + f(node->right, t-1);
}
```

- računanje broja elemenata na zadatom nivou stabla
- balansiranje stabla
- određivanje maksimalne dubine stabla
- pronalaženje najmanje vrednosti u stablu
- proverava da li postoji putanja od korena do lista sa zadatom sumom elemenata
- računanje broja elemenata u stablu
- računanje sume elemenata stabla
- zamenu levog i desnog podstabla, za svaki element u stablu

Od koliko polja se sastoji sledeća struktura:

```
struct osoba_st {
   char ime[50];
   char adresa[20];
   int godine;
};
```

- 3
- 1
- 2
- 4
- 71
- 74
- struktura nije pravilno deklarisana

Od koliko polja se sastoji sledeća struktura:

```
struct osoba_st {
   char ime[50];
   char adresa[20];
   int godine;
   int jmbg;
};
```

- 4
- 1
- 2
- 3

- 72
- 78
- struktura nije pravilno deklarisana

Od koliko polja se sastoji sledeća struktura:

```
struct {
   char ime[50];
   char adresa[20];
   int godine;
   int jmbg;
}
```

- struktura nije pravilno deklarisana
- 1
- 2
- 3
- 4
- 72
- 78

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


struct tacka_st {
   int x;
   int y;
};


int main() {
    struct tacka_st t;
    t.x = 3;
    t.y = 5;


    int i;
    for(i=0; i<t.x; i++){
        t.y += t.y;
    }


    printf("%d", t.y);


    return 0;
}
```

- 40
- 15
- 20
- greška prilikom kompajliranja: promenljiva `t` nije pravilno deklarisana
- greška prilikom kompajliranja: struktura nije pravilno deklarisana

---

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


struct tacka_st {
   int x;
   int y;
};


int main() {
    tacka_st t;
    t.x = 3;
    t.y = 5;

    int i;
    for(i=0; i<t.x; i++){
        t.y += t.y;
    }

    printf("%d", t.y);

    return 0;
}
```

- greška prilikom kompajliranja: promenljiva `t` nije pravilno deklarisana
- 15
- 20
- 40
- greška prilikom kompajliranja: struktura nije pravilno deklarisana

---

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


struct {
   int x;
   int y;
}
```

```
int main() {
    struct tacka_st t;
    t.x = 3;
    t.y = 5;

    int i;
    for(i=0; i<t.x; i++){
        t.y += t.y;
    }

    printf("%d", t.y);

    return 0;
}
```

- greška prilikom kompajliranja: struktura nije pravilno deklarisana
- 15
- 20
- 40
- greška prilikom kompajliranja: promenljiva `t` nije pravilno deklarisana

---

U sledećoj strukturi identifikator `tacka_st` je:

```
struct tacka_st {
    int x;
    int y;
};
```

- obavezan
- opcion
- u ovom slučaju struktura nije pravilno deklarisana

U sledećoj strukturi identifikator `tacka_st` je:

```
typedef struct tacka_st {
    int x;
    int y;
} TACKA;
```

- opcion
- obavezan
- u ovom slučaju `tacka_st` nije identifikator
- u ovom slučaju struktura nije pravilno deklarisana

U sledećoj strukturi identifikator `tacka_st` je:

```
typedef struct tacka_st {
    int x;
    int y;
    struct tacka_st *mama;
} TACKA;
```

- obavezan
- opcion
- u ovom slučaju `tacka_st` nije identifikator
- u ovom slučaju struktura nije pravilno deklarisana

U sledećoj strukturi identifikator `TACKA` je:

```
typedef struct tacka_st {
    int x;
    int y;
} TACKA;
```

- obavezan
- opcion
- u ovom slučaju `TACKA` nije identifikator
- u ovom slučaju struktura nije pravilno deklarisana

U sledećoj strukturi identifikator `TACKA` je:

```
typedef struct {
    int x;
    int y;
} TACKA;
```

- obavezan
- opcion
- u ovom slučaju `TACKA` nije identifikator
- u ovom slučaju struktura nije pravilno deklarisana

Kako se pravilno ispisuje polje `x`, koje se nalazi unutar promenljive `a`:

```
#include <stdio.h>


typedef struct tacka_st {
    int x;
    int y;
} TACKA;
```

```
int main() {
    TACKA a;

    TACKA *pa = &a;
    a.x = 1;
    a.y = 2;

    return 0;
}
```

- `printf("%d", pa->x);`
- `printf("%d", a->x);`
- `printf("%d", &a.x);`
- `printf("%d", pa.x);`
- `printf("%d", &pa.x);`

Kako se pravilno učitava polje `y`, koje se nalazi unutar promenljive `a`:

```
#include <stdio.h>

typedef struct tacka_st {
    int x;
    int y;
} TACKA;

int main() {
    TACKA a;

    TACKA *pa = &a;

    return 0;
}
```

- `scanf("%d", &pa->y);`
- `scanf("%d", &pa.y);`
- `scanf("%d", pa.y);`
- `scanf("%d", pa->y);`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

```c
#include <stdlib.h>

int main() {
    int n;
    printf("n = ");
    scanf("%d", &n);

    int **a;
    a = calloc(n, sizeof(int*));
    if(a == NULL) {
        puts("Greska 1 prilikom zauzimanja memorije!");
        exit(41);
    }

    int i, j;
    for (i=0; i<n; i++) {
        *(a+i) = calloc(n, sizeof(int));
        if(*(a+i) == NULL) {
            puts("Greska 2 prilikom zauzimanja memorije!");
            exit(42);
        }

        for (j=0; j<n; j++) {
            *(*(a+i)+j) = rand()/((double)RAND_MAX + 1) * 10;
        }
    }

    for (i=0; i<n; i++) {
        for (j=0; j<n; j++) {
            printf("%d ", *(*(a+i)+j));
        }

        printf("\n");
    }

    return 0;
}
```

- sadržaj matrice vrstu po vrstu
- sadržaj matrice vrstu po vrstu, a potom glavna dijagonala, sleva-udesno
- sadržaj matrice vrstu po vrstu, a potom sporedna dijagonala, sleva-udesno
- glavna dijagonala sdesna-ulevo
- glavna dijagonala sleva-udesno

- sporedna dijagonala sdesna-ulevo
- sporedna dijagonala sleva-udesno

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

int main() {
    int n;
    printf("n = ");
    scanf("%d", &n);

    int **a;
    a = calloc(n, sizeof(int*));
    if(a == NULL) {
        puts("Greska 1 prilikom zauzimanja memorije!");
        exit(41);
    }

    int i, j;
    for (i=0; i<n; i++) {
        *(a+i) = calloc(n, sizeof(int));
        if(*(a+i) == NULL) {
            puts("Greska 2 prilikom zauzimanja memorije!");
            exit(42);
        }

        for (j=0; j<n; j++) {
            *(*(a+i)+j) = rand()/((double)RAND_MAX + 1) * 10;
        }
    }

    for (i=0; i<n; i++) {
        for (j=0; j<n; j++) {
            printf("%d ", *(*(a+i)+j));
        }

        printf("\n");
    }

    for (i=n-1; i>=0; i--) {
```

```
        printf("%d ", *(*(a+i)+n-1-i));
    }
    printf("\n");

    return 0;
}
```

- sadržaj matrice vrstu po vrstu, a potom sporedna dijagonala, sleva-udesno
- sadržaj matrice vrstu po vrstu
- sadržaj matrice vrstu po vrstu, a potom glavna dijagonala, sleva-udesno
- glavna dijagonala sdesna-ulevo
- glavna dijagonala sleva-udesno
- sporedna dijagonala sdesna-ulevo
- sporedna dijagonala sleva-udesno

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

void f(int a) {
    a = 3;
}

int main() {
    int a = 5;
    f(a);

    printf("%i", a);

    return 0;
}
```

- 5
- 3
- prilikom kompajliranja dobijamo upozorenje da u pokazivač upisujemo ceo broj

---

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

void f(int *i) {
   *i = 3;
}
```

```
int main() {
   int a = 5;
   f(&a);

   printf("%d", a);

   return 0;
}
```

- 3
- 5
- prilikom kompajliranja dobijamo upozorenje da u pokazivač upisujemo ceo broj

---

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

void f(int *i) {
   i = 3;
}

int main() {
   int a = 5;
   f(&a);

   printf("%i", a);

   return 0;
}
```

- prilikom kompajliranja dobijamo upozorenje da u pokazivač upisujemo ceo broj
- 5
- 3

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

void f(int *i) {
    int a = 1;
    *i = 5;
}
```

```
int main() {
    int a = 2;
    int i = 3;

    printf("a = %d", a);

    f(&a);

    return 0;
}
```

- a = 2
- a = 1
- a = 5
- a = 3

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

void f(int *i) {
    int a = 1;
    *i = 5;
}

int main() {
    int a = 2;
    int i = a;
    f(&a);

    printf("a = %d", i);

    return 0;
}
```

- a = 2
- a = 1
- a = 5
- a = 3

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int f(int *i) {
```

```
    int b = 1;
    *i = 5;

    return b;
}

int main() {
    int a = 2;
    int i = 3;
    a = f(&a);

    printf("a = %d", a);

    return 0;
}
```

- a = 1
- a = 2
- a = 5
- a = 3

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int f(int *i) {
    int a = 1;
    *i = 5;

    return a;
}

int main() {
    int a = 2;
    int i = 3;

    printf("a = %d", a);

    a = f(&a);

    return 0;
}
```

- a = 2

- a = 1
- a = 5
- a = 3

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int f(int *i, int a) {
    a = *i;
    *i = 5;

    return a;
}

int main() {
    int a = 2;
    int i = 3;
    a = f(&i, a);

    printf("%d", a);

    return 0;
}
```

- 3
- 5
- 1
- 2

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int f(int *i, int a) {
    a = *i;
    *i = 5;

    return a;
}

int main() {
    int a = 2;
    int i = 3;
```

```
    a = f(&i, a);

    printf("%d", i);

    return 0;
}
```

- 5
- 3
- 1
- 2

Kako treba da izgleda deklaracija funkcije `unesi_niz`, koja nema povratnu vrednost i poziva se iz sledećeg dela koda:

```
#include <stdio.h>

int main() {
    unsigned n;
    unsigned niz[1000];

    unesi_niz(niz, &n);
    ispisi_niz(niz, n);

    return 0;
}
```

- `void unesi_niz(unsigned *,  unsigned *);`
- `void unesi_niz(unsigned *,  unsigned  );`
- `void unesi_niz(unsigned  ,  unsigned *);`
- `void unesi_niz(unsigned  , &unsigned  );`
- `int  unesi_niz(unsigned *,  unsigned *);`

Kako treba da izgleda deklaracija funkcije `ispisi_niz`, koja nema povratnu vrednost i poziva se iz sledećeg dela koda:

```
#include <stdio.h>

int main() {
    unsigned n;
    unsigned niz[1000];

    unesi_niz(niz, &n);
```

```
    ispisi_niz(niz, n);


    return 0;
}
```

- `void ispisi_niz(unsigned *,  unsigned  );`
- `void ispisi_niz(unsigned *,  unsigned *);`
- `void ispisi_niz(unsigned  ,  unsigned *);`
- `void ispisi_niz(unsigned  , &unsigned  );`
- `int  ispisi_niz(unsigned *,  unsigned *);`

Kog je tipa povratna vrednost sledeće funkcije:

```
void zbir(int x, int y, int *k) {
    *k = x + y;
}
```

- ova funkcija nema povratnu vrednost
- `int`
- `int *`
- `float`

Kog je tipa povratna vrednost sledeće funkcije:

```
int *zbir(int x, int y, int *k) {
    *k = x + y;
}
```

- `int *`
- `int`
- `float`
- ova funkcija nema povratnu vrednost

Nizovi i pokazivači

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>


int main() {
    int i = 0;
    int a[10] = {0, 1, 2, 3, 4, 5, 6, 8, 9, 0};


    int *p = a;
    while(*p) {
        printf("%d", i);
```

```
        p++;
        i++;
    }

    return 0;
}
```

- 0 puta
- 1 put
- 10 puta
- 11 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;
    int a[10] = {-2, -1, 0, 1, 2, 3, 4, 5, 8, 0};

    int *p = a;
    while(*p) {
        printf("%d", i);

        p++;
        i++;
    }

    return 0;
}
```

- 2 puta
- 0 puta
- 1 put
- 10 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

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

```
int main() {
    int i = 0;
    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};

    int *p = a;
    while(*p) {
        printf("%d", i);

        p++;
        i++;
    }

    return 0;
}
```

- 9 puta
- 0 puta
- 1 put
- 10 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;
    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0};

    int *p = a;
    while(*p) {
        printf("%d", i);

        p++;
        i++;
    }

    return 0;
}
```

- 10 puta
- 0 puta
- 9 puta

- 11 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>


int main() {
    int i = 0;
    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0};


    int *p = a + 1;
    while(*p){
        printf("%d", i);


        p++;
        i++;
    }


    return 0;
}
```

- 9 puta
- 0 puta
- 10 puta
- 11 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>


int main() {
    int i = 0;
    int a[10] = {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};


    int *p = a;
    while(*p-4){
        printf("%d", i);


        p++;
        i++;
    }
```

```
    return 0;
}
```

- 5 puta
- 0 puta
- 6 puta
- 9 puta
- 11 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;
    int a[10] = {9, 8, 7, 6, 5, 4, 3, 2, 0};

    int *p = a;
    while(*(p+4)) {
        printf("%d", i);

        p++;
        i++;
    }

    return 0;
}
```

- 4 puta
- 0 puta
- 5 puta
- 6 puta
- 9 puta
- 11 puta
- beskonačno mnogo puta

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i = 0;
    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
```

```
    int *p = a;
    while(*p){
        p++;
        i--;
    }

    printf("%d", i);

    return 0;
}
```

- 1
- 0
- 5
- 6
- 9
- 11
- ništa neće biti ispisano, zbog beskonačne petlje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i = 0;
    int j = 2;
    int a[7] = {0, 1, 2, 3, 2, 1, 0};

    while(a+i != a+j) i++; j--; printf("%d %d ", a[i], a[j]);

    return 0;
}
```

- 2 1
- 0 0
- 0 2
- 1 0
- 2 3
- program će prilikom izvršavanja ući u beskonačnu petlju

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

```
int main() {
    int i = 0;
    int j = 2;
    int a[7] = {0, 1, 2, 3, 2, 1, 0};

    while(a+i != a+j) i++; j++; printf("%d %d ", a[i], a[j]);

    return 0;
}
```

- 2 3
- 0 1
- 0 2
- 1 2
- 2 1
- program će prilikom izvršavanja ući u beskonačnu petlju

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i = 0;
    int j = 2;
    int a[7] = {0, 1, 2, 3, 2, 1, 0};

    while(a+i == a+j) i++; j++; printf("%d %d ", a[i], a[j]);

    return 0;
}
```

- 0 3
- 0 1
- 1 2
- 2 1
- 2 3
- program će prilikom izvršavanja ući u beskonačnu petlju

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i = 10;
    int a[10] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 0};
```

```
    int *p = a;
    while(*p) {
        p += *p;
        i--;
    }

    printf("%d", i);

    return 0;
}
```

- 1
- 0
- 3
- 6
- 9
- 11
- program će prilikom izvršavanja ući u beskonačnu petlju

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i;
    int a[10] = {2, 1, 10, 7, 10, 0, 8, 3, 9, 5};

    int *p = a + 2;
    for(i=0; i<10; i++)
        a[i] = i + 2;

    printf("%d", *p);

    return 0;
}
```

- 4
- 0
- 1
- 2
- 10
- program će prilikom izvršavanja ući u beskonačnu petlju

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>


int main() {
    int i = 5;
    int a[10] = {-4, -2, 0, 1, 2, 3, 4, 5, 8, 0};


    int *p = a + 2;
    do {
        printf("%d", i);


        p++;
        i--;
    } while(*p);


    return 0;
}
```

- 7 puta
- 0 puta
- 1 put
- 2 puta
- 5 puta
- 8 puta
- 10 puta
- beskonačno mnogo puta

Nizovi

Odabrati ispravan `USLOV`:

```
do {
    printf("Unesite broj elemenata");
    scanf("%d", &n);
} while( USLOV );
```

- `n <= 0 || n >  MAX_SIZE`
- `n <  0 || n >  MAX_SIZE`
- `n <  0 || n >= MAX_SIZE`
- `n >= 0 || n <  MAX_SIZE`
- `n <= 0 && n >  MAX_SIZE`
- `n <  0 && n >  MAX_SIZE`
- `n <  0 && n >= MAX_SIZE`
- `n >= 0 && n <  MAX_SIZE`

Kako se može deklarisati niz celobrojnih vrednosti sa jednim članom?

- `int niz[1];`
- `int niz(1);`
- `int niz{1};`
- nije moguće deklarisati niz sa jednim članom

Kako se pravilno deklariše celobrojni niz `a`?

- `int a[5];`
- `int [5]a;`
- `int a;[5];`
- `int [a:5];`
- `int []a;`
- `int a(5);`
- `int a{5};`

Kako se pravilno učitava drugi član celobrojnog niza `a`?

- `scanf("%d", &a[1]);`
- `scanf(&a[2]);`
- `scanf("%d", %a[2]);`
- `scanf("%d", a[1]);`
- `scanf("%d", &a+1);`
- `scanf("%d", *a+1);`
- `scanf("%d", a+2);`
- `scanf("%d", &a[2]);`
- `scanf("%d", &(a+1));`
- `scanf("d", &a[2]);`

Kako se pravilno učitava prvi član celobrojnog niza `a`?

- `scanf("%d", &a[0]);`
- `scanf(&a[1]);`
- `scanf("%d", %a[1]);`
- `scanf("%d", a[0]);`
- `scanf("%d", &a);`
- `scanf("%d", *a);`
- `scanf("%d", a+1);`
- `scanf("%d", &a[1]);`
- `scanf("%d", &(a+1));`
- `scanf("d", &a[1]);`

Odabrati ispravan `SNIPPET` za ispis niza u obrnutom redosledu:

```
for (SNIPPET) {
```

```
    printf("a[%d] = %d\n", j, a[j]);
}
```

- `j = n-1; j >= 0; j--`
- `i = n-1; i >= 0; i--`
- `j = n-1; j > 0; j--`
- `i = n-1; i > 0; i--`
- `j = n; j >= 0; j--`
- `i = n; i >= 0; i--`
- `j = n; j > 0; j--`
- `i = n; i > 0; i--`
- `j = n-1; j > 0; j++`
- `i = n-1; i > 0; i++`

Odabrati ispravan `SNIPPET` za ispis svakog 3. člana niza:

```
for (SNIPPET) {
    printf("a[%d] = %d\n", j, a[j]);
}
```

- `j = 0; j < n; j+=3`
- `i = 0; i < n; i+=3`
- `i = 0; i < n; i++3`
- `i = 0; i < n; i+3`
- `j = 0; j < n; j+3`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i=1;
    int a[10]={-2,-1,0,1,2,3,4,5,8,0};

    printf("%d", a[i]);

    return 0;
}
```

- -1
- -2
- 0
- 1
- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i=4;
    int a[10]={-2,-1,0,1,2,3,4,5,8,0};

    printf("%d", a[i]);

    return 0;
}
```

- 2
- -1
- 0
- 1
- 3
- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i;
    int a[10];

    for (i=0;i<10;i++){
        a[i]=0;
        if(i==2 || i==4)
            a[i]=5;
    }

    printf("%d", a[0]+a[1]+a[2]);

    return 0;
}
```

- 5
- 0
- 10
- 1
- 15
- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i;
    int a[10];

    for (i=0;i<10;i++){
        a[i]=0;
        if(i<2)
            a[i]=5;
    }

    printf("%d", a[0]+a[1]+a[2]);

    return 0;
}
```

- 10
- 0
- 5
- 1
- 15
- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i, a[10];

    for(i=0;i<10;i++)
        a[i]=i+2;

    printf("%d", a[0]+a[5]);

    return 0;
}
```

- 9
- 0
- 2
- 5

- 7

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a[10];


    printf("%d", a[0]);


    return 0;
}
```

- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)
- 0
- 1
- 10

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a[5];
    for (i=2;i<5;i++)
        a[i]=5;


    printf("%d", a[0]);


    return 0;
}
```

- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)
- 0
- 1
- 5
- 10

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int i;
```

```
    int a[5];
    int suma=0;

    for (i=2;i<5;i++)
        a[i]=5;
        suma += a[i];

    printf("%d", suma);

    return 0;
}
```

- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)
- 5
- 10
- 15
- 25

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i = 0;
    int j = 2;
    int a[7] = {0, 1, 2, 3, 2, 1, 0};

    while(a[i] != a[j]) i++; j++; printf("%d %d ", a[i], a[j]);

    return 0;
}
```

- 2 3
- 0
- 1
- 2
- 0 2 1 3
- program će prilikom izvršavanja ući u beskonačnu petlju

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i = 0;
```

```
    int j = 2;
    int a[7] = {0, 1, 2, 3, 2, 1, 0};

    while(a[i] = a[j]) i++; j++; printf("%d %d ", a[i], a[j]);

    return 0;
}
```

- program će prilikom izvršavanja ući u beskonačnu petlju
- 2 1
- 2 3
- 0 3
- 0 1
- 1 2

Petlje

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;

    for(i=5; i<=10; i++)
        printf("%d", i);

    return 0;
}
```

- 6 puta
- 0 puta
- 4 puta
- 5 puta
- 9 puta
- 10 puta
- 11 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;
```

```
    for(i=5; i>=10; i++)
        printf("%d", i);

    return 0;
}
```

- 0 puta
- 4 puta
- 5 puta
- 6 puta
- 9 puta
- 10 puta
- 11 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;

    for(i==5; i<=10; i++)
        printf("%d", i);

    return 0;
}
```

- 11 puta
- 0 puta
- 4 puta
- 5 puta
- 6 puta
- 9 puta
- 10 puta
- beskonačno mnogo puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;
```

```
    for(i=5; i=10; i++)
        printf("%d", i);

    return 0;
}
```

- beskonačno mnogo puta
- 0 puta
- 4 puta
- 5 puta
- 6 puta
- 9 puta
- 10 puta
- 11 puta

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;

    for(i=0; i; i++)
        printf("%d", i);

    return 0;
}
```

- 0 puta
- 4 puta
- 5 puta
- 6 puta
- 9 puta
- 10 puta
- 11 puta
- beskonačno mnogo puta

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i = 0;
    int a = 0;
```

```
    for(i=5; i<=10; i+=2)
        a++;

    printf("%d", a);

    return 0;
}
```

- 3
- 0
- 2
- 4
- 5
- 6
- ništa neće biti ispisano, zbog beskonačne petlje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i = 0;
    int a = 0;

    for(i=10; i>=5; i-=2)
        a++;

    printf("%d", a);

    return 0;
}
```

- 3
- 0
- 2
- 4
- 5
- 6
- ništa neće biti ispisano, zbog beskonačne petlje

Nakon izvršavanja sledećeg dela koda vrednost promenljive `i` će biti:

```
int a = 6;
int i = 9;

do {
```

```
    i++;
} while(a < 5);
```

- 10
- 6
- 7
- 9
- 11

Nakon izvršavanja sledećeg dela koda vrednost promenljive `a` će biti:

```
int a = 9;
int i = 6;

do {
    i++;
} while(a < 5);

printf("%d", i);
```

- 9
- 6
- 7
- 10
- 11

Nakon izvršavanja sledećeg dela koda vrednost promenljive `i` će biti:

```
int a = 3;
int i = 6;

do {
    i++;
    a--;
} while(a > 5 || i < 8);
```

- 8
- 2
- 3
- 6
- 7

Nakon izvršavanja sledećeg dela koda vrednost promenljive `a` će biti:

```
int a = 3;
int i = 6;
```

```
do {
    i++;
    a--;
} while(a > 5 || i < 8);
```

- 1
- 0
- 3
- 6
- 7

Nakon izvršavanja sledećeg dela koda vrednost promenljive `i` će biti:

```
int a = 6;
int i = 9;

while(a > 5){
    i = --a;
}
```

- 5
- 4
- 6
- 7

Nakon izvršavanja sledećeg dela koda vrednost promenljive `i` će biti:

```
int a = 7;
int i = -5;

do {
    i = --a;
} while(a >= 5);

printf("%d", i);
```

- 4
- -5
- 5
- 6
- 7

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

```
int main() {
    int i = 0;
    int a = 5;

    while(a >= 3) {
        a--;
        i++;
    }

    printf("%d", i);

    return 0;
}
```

- 3
- 0
- 1
- 2
- 4

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = 0;
    int a = 5;

    while(a)
        i++;

    return 0;
}
```

- beskonačno mnogo puta
- 0 puta
- 1 put
- 2 puta
- 5 puta

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

```
int main() {
    int i = 5;
    int a = 0;

    while(a)
        i++;

    printf("%d", i);

    return 0;
}
```

- 5
- 0
- 1
- 6
- ništa neće biti ispisano, zbog beskonačne petlje

Koliko puta će se izvršiti telo petlje u sledećem kodu:

```
#include <stdio.h>

int main() {
    int i = -3;
    int a = 4;
    int p = 0;

    while(p = i-a+2) {
        printf("%d", i);
        i++;
    }

    return 0;
}
```

- 5 puta
- 0 puta
- 1 put
- 6 puta
- beskonačno mnogo puta

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

```
int main() {
    int i;

    for(i=1; i<=5; i++) {
        if(i == 3) {
            break;
            continue;
        }

        printf("%d ", i);
    }

    return 0;
}
```

- 1 2
- 1 2 4 5
- 0 1 2
- 0 1 2 3 4 5
- 1 2 3 4 5
- 0 1 2 4 5
- program će prilikom izvršavanja ući u beskonačnu petlju

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int i;

    for(i=1; i<=5; i++) {
        if(i == 3) {
            continue;
            break;
        }

        printf("%d ", i);
    }

    return 0;
}
```

- 1 2 4 5
- 1 2
- 0 1 2

- `0 1 2 3 4 5`
- `1 2 3 4 5`
- `0 1 2 4 5`
- program će prilikom izvršavanja ući u beskonačnu petlju

Ako se uporedi data `while` petlja:

```
while(i < j) {
    i = 1;
    x--;
    i++;
}
```

sa sledećom `for` petljom:

```
for(i=1; i<j; i++) {
    x--;
}
```

može se tvrditi da:

- ove dve petlje nisu ekvivalentne
- ove dve petlje jesu ekvivalentne
- `for` petlja nije sintaksno ispravna
- `while` petlja nije sintaksno ispravna

Ako se uporedi data `while` petlja:

```
i = 1;
while(i < j) {
    x--;
    i++;
}
```

sa sledećom `for` petljom:

```
for(i=1; i<j; i++) {
    x--;
}
```

može se tvrditi da:

- ove dve petlje jesu ekvivalentne
- ove dve petlje nisu ekvivalentne
- `for` petlja nije sintaksno ispravna
- `while` petlja nije sintaksno ispravna

Koliko puta će se izvršiti telo petlje u sledećem kodu:

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

```
int main() {
    int j;

    for(j=273; j>732; j++)
        printf("PJISP ");

    return 0;
}
```

- 0 puta
- 1 put
- 459 puta
- 460 puta
- beskonačno mnogo puta

Koliko puta će se ispisati reč "PJISP" na standardni izlaz, kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int x = -1;

    for( ; x<=10; x++) {
        if(x < 5)
            continue;
        else
            break;

        printf("PJISP ");
    }

    return 0;
}
```

- 0 puta
- 9 puta
- 10 puta
- 11 puta
- beskonačno mnogo puta

Šta od navedenog ne može da bude uslov `while` petlje, ukoliko je `lower` promenljiva tipa `int`?

- `if(lower <= 1)`

- `lower <= 1`
- `lower + 1`
- `lower++`
- svi navedeni primeri mogu biti uslov `while` petlje

Odabrati ispravan `USLOV`, kojim će se reč "PJISP" ispisati tačno 15 puta na standardni izlaz:

```
#include <stdio.h>


int main() {
    int j;


    for(j=2; USLOV ; j++)
        printf("PJISP ");


    return 0;
}
```

- `j <= 16`
- `j > 15`
- `j < 16`
- `j == 16`
- `j <= 17`

Stringovi

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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


int main() {
    char a[]={'1','2','3','\0','4','\0'};


    printf("%d", strlen(a));


    return 0;
}
```

- 3
- 5
- 4
- 6

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

int main() {
    char a[]={'1','2','3','0','4','\0'};

    printf("%d", strlen(a));

    return 0;
}
```

- 5
- 4
- 6
- 3

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

int main() {
    char a[]={'0','1','2','3','\0','4','0'};

    printf("%d", strlen(a));

    return 0;
}
```

- 4
- 5
- 6
- 3

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

int main() {
    char a[]={'\0','1','2','3','0','4','\0'};

    printf("%d", strlen(a));
```

```
    return 0;
}
```

- 0
- 5
- 4
- 6
- 3

Tipovi

U programskom jeziku C konstante i promenljive mogu biti tipa `void`.

- netačno
- tačno

Kog tipa će biti sledeći izraz:

```
(short int)3/3.0
```

- `double`
- `int`
- `char`
- `short int`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, smatrati da se kod izvršava na 32-bitnoj arhitekturi:

```
#include <stdio.h>


int main() {
    printf("%d", sizeof(float));


    return 0;
}
```

- 4
- float
- d
- %d
- %d32
- 2
- 16
- 32

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda, smatrati da se kod izvršava na 32-bitnoj arhitekturi:

```
#include <stdio.h>


int main() {
    printf("%d", sizeof(int));


    return 0;
}
```

- 4
- int
- d
- %d
- %d32
- 2
- 16
- 32

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 5;
    int b = 3;


    printf("Realni kolicnik a/b je: %d\n", a/b);


    return 0;
}
```

- `Realni kolicnik a/b je: 1`
- `Realni kolicnik a/b je: 2`
- `Realni kolicnik a/b je: 1.000000`
- `Realni kolicnik a/b je: 1.666667`
- `Realni kolicnik a/b je: 2.000000`
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 5;
```

```
    float b = 3;

    printf("Celobrojni kolicnik a/b je: %f\n", a/b);

    return 0;
}
```

- Celobrojni kolicnik a/b je: 1.666667
- Celobrojni kolicnik a/b je: 1
- Celobrojni kolicnik a/b je: 2
- Celobrojni kolicnik a/b je: 1.000000
- Celobrojni kolicnik a/b je: 2.000000
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int a = 5;
    int b = 3;

    printf("Celobrojni kolicnik a/b je: %d\n", a%b);

    return 0;
}
```

- Celobrojni kolicnik a/b je: 2
- Celobrojni kolicnik a/b je: %d
- Celobrojni kolicnik a/b je: 1
- Celobrojni kolicnik a/b je: 1.000000
- Celobrojni kolicnik a/b je: 1.666667
- Celobrojni kolicnik a/b je: 2.000000
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int a = 5;
    float b = 3;

    printf("Celobrojni kolicnik a/b je: %%d\n", a/b);
```

```
    return 0;
}
```

- `Celobrojni kolicnik a/b je: %d`
- `Celobrojni kolicnik a/b je: 1`
- `Celobrojni kolicnik a/b je: 2`
- `Celobrojni kolicnik a/b je: 1.000000`
- `Celobrojni kolicnik a/b je: 1.666667`
- `Celobrojni kolicnik a/b je: 2.000000`
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    float avr;
    int suma=10, br=3;
    avr = suma/br;


    printf("%f\n", avr);


    return 0;
}
```

- 3.000000
- 3.333333
- 3
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    float avr;
    int suma=10, br=3;
    avr = suma/3.0;


    printf("%f\n", avr);


    return 0;
}
```

- 3.333333
- 3.000000

- 3
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    float avr;
    int suma=10, br=3;
    avr = (float)suma/br;


    printf("%d\n", 3);


    return 0;
}
```

- 3
- 3.333333
- 3.000000
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    float avr;
    int suma=10, br=3;
    avr = (float)suma/br;


    printf("%d\n", avr);


    return 0;
}
```

- prilikom kompajliranja dobijamo grešku/upozorenje
- 3
- 3.333333
- 3.000000

Nakon izvršavanja sledećeg dela koda vrednost promenljive `fahr` će biti:

```
int celsius = 1;
double fahr;
```

```
fahr = celsius / 5 * 9.0 + 32.0;
```

- 32.0
- 33.8
- neodređena vrednost

Nakon izvršavanja sledećeg dela koda vrednost promenljive `fahr` će biti:

```
int celsius;
double fahr = 1;
fahr = celsius / 5 * 9.0 + 32.0;
```

- neodređena vrednost
- 32.0
- 33.8

Nakon izvršavanja sledećeg dela koda vrednost promenljive `x` će biti:

```
int a;
int b = 1;
int x;
x = a * (a % b) + 10;
```

- 10
- 11
- 0
- 13
- neodređena vrednost

Nakon izvršavanja sledećeg dela koda vrednost promenljive `x` će biti:

```
int a;
int b = 3;
int x;
x = a * (a % b) + 10;
```

- neodređena vrednost
- 11
- 0
- 13
- 10

Koji od navedenih tipova dozvoljava najvišu numeričku vrednost?

- `double`
- `char`
- `float`

- `int`
- `short`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int i, j=0, p=1, q=2;


    printf("%d", i*j);


    return 0;
}
```

- 0
- 1
- 2
- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int i, j=0, p=1, q=2;


    printf("%d", i);


    return 0;
}
```

- ne možemo znati koji broj će biti ispisan (vrednost zatečena u memoriji)
- 0
- 1
- 2
- prilikom kompajliranja dobijamo grešku/upozorenje

Uslovi

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

```
int main() {
    int a = 2;
    int b = 1;

    if (a = b)
        printf("Veci je b");
    else
        printf("Veci je a");

    return 0;
}
```

- Veci je b
- Veci je a

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int a = 1;
    int b = 2;

    if (a==b)
        printf("Veci je b");
    else
        printf("Veci je a");

    return 0;
}
```

- Veci je a
- Veci je b

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int a = 2;
    int b = 1;

    if (a > b)
        printf("Veci je b");
```

```
    else
        printf("Veci je a");

    return 0;
}
```

- Veci je b
- Veci je a

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int a = 1;
    int b = 0;

    if (a ? b : a)
        printf("Veci je b");
    else
        printf("Veci je a");

    return 0;
}
```

- Veci je a
- Veci je b

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int a = 0;
    int b = 1;

    if ((a > b) ? a : b)
        printf("Veci je b");
    else
        printf("Veci je a");

    return 0;
}
```

- Veci je b

- Veci je a

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int ocena = 1;

    switch (ocena) {
        case 5:  printf("Odlican!");
        case 4:  printf("Vrlo dobar!");
        case 3:  printf("Dobar!");
        case 2:  printf("Dovoljan!");
        case 1:  printf("Nedovoljan!");
        default: printf("Ocena mora biti izmedju 1 i 5.");
    }

    return 0;
}
```

- Nedovoljan!Ocena mora biti izmedju 1 i 5.
- Nedovoljan!
- Ocena mora biti izmedju 1 i 5.
- ništa neće biti ispisano

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int ocena = 3;

    switch (ocena) {
        case 5:  break;printf("Odlican!");
        case 4:  break;printf("Vrlo dobar!");
        case 3:  break;printf("Dobar!");
        case 2:  break;printf("Dovoljan!");
        case 1:  break;printf("Nedovoljan!");
        default: break;printf("Ocena mora biti izmedju 1 i 5.");
    }

    return 0;
```

```
}
```

- ništa neće biti ispisano
- `Dobar!Ocena mora biti izmedju 1 i 5.`
- `Dobar!`
- `Ocena mora biti izmedju 1 i 5.`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int a = 1;
    int b = 0;

    if (!b)
        printf("Jednaki! ");
    else
        printf("Razliciti! ");
    printf("%d\n", b);

    return 0;
}
```

- `Jednaki! 0`
- `Razliciti! 1`
- `Razliciti! 0`
- `Jednaki! 1`

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    int a = 1;
    int b = 0;

    if (b = a)
        printf("Jednaki! ");
    else
        printf("Razliciti! ");
    printf("%d\n", b);

    return 0;
}
```

- Jednaki! 1
- Razliciti! 1
- Razliciti! 0
- Jednaki! 0

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 1;
    int b = 0;

    if (a = b)
        printf("Jednaki! ");
    else
        printf("Razliciti! ");
    printf("%d\n", a);

    return 0;
}
```

- Razliciti! 0
- Jednaki! 1
- Razliciti! 1
- Jednaki! 0

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 5;
    int b = 3;

    if ((b > a) || (b = a))
        printf("Manje! ");
    else
        printf("Vece! ");
        printf("Razlicito! ");

    return 0;
}
```

- Manje! Razlicito!

- Manje!
- Vece! Razlicito!
- Razlicito!
- Vece!

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 5;
    int b = 3;

    if ((b > a) || (b == a))
        printf("Manje! ");
    else
        printf("Vece! ");
        printf("Razlicito! ");

    return 0;
}
```

- Vece! Razlicito!
- Manje!
- Razlicito!
- Vece!
- Manje! Razlicito!

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 5;
    int b = 3;

    if (b = a)
        printf("Manje! ");
    else if (b < a)
        printf("Vece! ");
        printf("Razlicito! ");

    return 0;
}
```

- Manje! Razlicito!
- Manje!
- Vece! Razlicito!
- Razlicito!
- Vece!

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int x = -3;


    if (x + 3)
        printf("istina");
    else
        printf("neistina");


    return 0;
}
```

- neistina
- istina
- istinaneistina
- ništa neće biti ispisano
- prilikom kompajliranja dobijamo grešku/upozorenje

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg dela koda:

```
int x = 10, y = 20, z = 30;
if(x == 10) x = 20; y = 30; z = 40;
if(y == 20) x = 30; y = 40; z = 50;
if(z == 50) x = 10; y = 20; z = 30;


printf("%d %d %d", x, y, z);
```

- 10 20 30
- 30 40 50
- 20 30 40

Znakovi

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

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

```
int main() {
    // 'j' == 106
    // 'k' == 107

    char promenljiva_1 = 'j';
    int promenljiva_2;
    int promenljiva_3 = 1;

    printf("%c", promenljiva_1 + promenljiva_3);

    return 0;
}
```

- k
- j
- 106
- 107
- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    // 'j' == 106
    // 'k' == 107

    char promenljiva_1 = 'j';
    int promenljiva_2;
    int promenljiva_3 = 1;

    printf("%d", promenljiva_1 + promenljiva_3);

    return 0;
}
```

- 107
- j
- k
- 106
- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```c
#include <stdio.h>

int main() {
    // 'j' == 106
    // 'k' == 107

    char promenljiva_1 = 'j';
    int promenljiva_2;
    int promenljiva_3 = 1;

    printf("%d", promenljiva_1 + promenljiva_2);

    return 0;
}
```

- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)
- j
- k
- 106
- 107

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```c
#include <stdio.h>

int main() {
    // 'J' ==  74
    // 'K' ==  75
    // 'j' == 106
    // 'k' == 107

    char promenljiva_1 = 'j';
    int promenljiva_2 = -32;
    int promenljiva_3 = 1;

    printf("%c", promenljiva_1 + promenljiva_2 + promenljiva_3);

    return 0;
}
```

- K
- J
- j
- k
- 106

- 107
- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    // 'J' ==  74
    // 'K' ==  75
    // 'j' == 106
    // 'k' == 107

    char promenljiva_1 = 'j';
    int promenljiva_2 = -32;
    int promenljiva_3 = 1;

    printf("%c", promenljiva_1 + promenljiva_2);

    return 0;
}
```

- J
- K
- j
- k
- 106
- 107
- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    // '1' ==  49
    // '2' ==  50
    // 'b' ==  98

    char promenljiva_1 = '1';
    int promenljiva_2;
    int promenljiva_3 = 1;

    printf("%c", promenljiva_1 + promenljiva_3);
```

```
    return 0;
}
```

- 2
- 1
- 11
- b
- 49
- 50
- 98
- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    // '1' ==  49
    // '2' ==  50
    // 'b' ==  98

    char promenljiva_1 = '1';
    int promenljiva_2;
    int promenljiva_3 = '1';

    printf("%c", promenljiva_1 + promenljiva_3);

    return 0;
}
```

- b
- 1
- 2
- 11
- 49
- 50
- 98
- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    // '1' ==  49
```

```
    // '2' ==  50
    // 'b' ==  98

    char promenljiva_1 = '1';
    int promenljiva_2;
    int promenljiva_3 = '1';

    printf("%d", promenljiva_1 + promenljiva_3);

    return 0;
}
```

- 98
- 1
- 2
- 11
- b
- 49
- 50
- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>

int main() {
    // '1' ==  49
    // '2' ==  50
    // 'b' ==  98

    char promenljiva_1 = '1';
    int promenljiva_2;
    int promenljiva_3 = '1';

    printf("%c", promenljiva_1 + promenljiva_2);

    return 0;
}
```

- ne možemo znati koji podatak će biti ispisan (vrednost zatečena u memoriji)
- 1
- 2
- 11
- b
- 49

- 50
- 98

Šta će biti ispisano na standardni izlaz kao rezultat izvršavanja sledećeg koda:

```
#include <stdio.h>


int main() {
    int a = 1;
    int b = 2;


    if (&a == &b)
        printf("Veci je b");
    else
        printf("Veci je a");


    return 0;
}
```

- Veci je a
- Veci je b
- prilikom kompajliranja dobijamo grešku/upozorenje (ne možemo porediti adrese!)