

Napomena: u svakom zadatku u kojem se traži definiranje funkcije, potrebno je napisati i prikladnu funkciju *main* ("glavni program") pomoću kojeg funkciju treba testirati.

1. Napisati funkciju naziva *kvadrat*, tipa *int*, koja za zadani cijeli broj *n* (parametar je tipa *int*) vraća kvadrat broja *n*, n^2 . U glavnom programu s tipkovnice učitati cijeli broj i ispisati rezultat.

Primjer izvršavanja programa.

```
Upisite cijeli broj > -6↵
-6 na kvadrat jest 36
```

2. Provjeriti hoće li se dobiti ispravan rezultat kada se pomoću funkcije *kvadrat* iz 1. zadatka pokuša izračunati -50000^2 . Objasniti što se dogodilo. Prepraviti obje funkcije (*kvadrat* i *main*) iz prethodnog zadatka tako da se može ispravno izračunati -50000^2 .
3. Provjeriti hoće li se dobiti ispravan rezultat kada se pomoću funkcije *kvadrat* iz 1. zadatka pokuša izračunati 3.5^2 . Objasniti što se dogodilo. Prepraviti obje funkcije (*kvadrat* i *main*) iz 1. zadatka tako da se može ispravno izračunati 3.5^2 .

Primjer izvršavanja programa.

```
Upisite realni broj > 3.5↵
3.500000 na kvadrat jest 12.250000
```

4. Napisati funkciju naziva *jestVelikoSlovo* koja za zadani znak vraća logičku vrijednost istina ili laž, ovisno o tome je li zadani znak veliko slovo ili nije. Sami odredite odgovarajući tip funkcije, broj i tip parametara. U glavnom programu s tipkovnice učitati jedan znak te na zaslon, ovisno o rezultatu funkcije, ispisati poruku "Jest veliko slovo" ili poruku "Nije veliko slovo".

Primjer izvršavanja programa.

```
Upisite znak > B↵
Jest veliko slovo
```

5. Napisati funkciju naziva *tablica* koja na zaslon ispisuje tablicu množenja za zadanih *m* redaka i *n* stupaca. Sami odredite odgovarajući tip funkcije, broj i tip parametara.

Primjer izvršavanja programa koji funkciju pozove s argumentima 3, 4.

```
Upisite broj redaka i stupaca > 3 4↵
.....1....2....3....4↵
....1....1....2....3....4↵
....2....2....4....6....8↵
....3....3....6....9...12↵
```

6. Napisati funkciju naziva *fibonacci15* koja na zaslon, u jednom retku, odijeljene zarezima, ispisuje prvih 15 Fibonaccijevih brojeva. Sami odredite odgovarajući tip funkcije, broj i tip parametara.

Primjer izvršavanja programa koji pozove funkciju *fibonacci15*

```
1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610
```

7. Napisati funkciju naziva `fibonacciN` koja na zaslon, u jednom retku, odijeljene zarezima, ispisuje zadani broj Fibonaccijevih brojeva (zadani broj članova može biti nenegativni broj, dakle i nula). Sami odredite odgovarajući tip funkcije, broj i tip parametara. U glavnom programu učitati cijeli broj (željeni broj članova koje treba ispisati) i pozivom funkcije ispisati članove niza.

Primjeri izvršavanja programa

```
Upisite broj clanova > 10↵  
1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

```
Upisite broj clanova > 2↵  
1, 1
```

```
Upisite broj clanova > 1↵  
1
```

```
Upisite broj clanova > 0↵
```

8. Napisati funkciju naziva `kolikiJeInt` koja vraća broj bajtova koji se koriste za pohranu podatka tipa `int`. Napomena: različiti prevodioci mogu za pohranu podatka tipa `int` koristiti različiti broj bajtova, te se funkcija koja vraća *konstantu* 4 (npr. pomoću `return 4;`) ne može smatrati ispravnom (prenosivom, portabilnom).
9. Napisati funkciju `getBit` koja vraća vrijednost `n`-tog bita zadane vrijednosti `x` tipa `unsigned int`. U funkciji `main` učitati vrijednost nenegativnog cijelog broja, vrijednost za `n` (redni broj bita), pozivom funkcije izračunati vrijednost `n`-tog bita, te ga ispisati.

Primjeri izvršavanja programa.

```
Upisite nenegativni cijeli broj > 29↵  
Upisite redni broj bita > 4↵  
Vrijednost bita je 1
```

```
Upisite nenegativni cijeli broj > 5↵  
Upisite redni broj bita > 0↵  
Vrijednost bita je 1
```

10. Napisati funkciju `printBinary` koja će za zadanu vrijednost nenegativnog cijelog broja, pomoću 32 uzastopna poziva funkcije `getBit`, na zaslon ispisati njegovu binarnu vrijednost. U funkciji `main` učitati vrijednost za nenegativni cijeli broj te pozivom funkcije `printBinary` ispisati njegov binarni ekvivalent.

Primjeri izvršavanja programa.

```
Upisite nenegativni cijeli broj > 29↵  
000000000000000000000000000011101
```

```
Upisite nenegativni cijeli broj > 0↵  
00000000000000000000000000000000
```

```
Upisite nenegativni cijeli broj > 2147483650↵  
100000000000000000000000000000010
```

11. Napisati funkciju naziva sinus koja će za kut x izražen u radijanima izračunati približnu vrijednost funkcije sinus kao parcijalnu sumu n članova niza. Funkcija kao argumente prima realni broj x i pozitivan cijeli broj n koji predstavlja broj članova niza koje treba sumirati. U glavnom programu učitati realni broj x i pozitivni cijeli broj n , izračunati parcijalnu sumu niza i ispisati koliko rezultat dobiven parcijalnom sumom odstupa od rezultata koji bi se dobio korištenjem funkcije `sin` iz `<math.h>`. U funkciji i glavnom programu koristiti realne brojeve dvostruke preciznosti i cijele brojeve najvećeg mogućeg raspona (tamo gdje je potrebno radi faktorijela).

$$\sin(x) \approx \sum_{i=1}^n (-1)^{i+1} \frac{x^{2i-1}}{(2i-1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots + (-1)^{n+1} \frac{x^{2i-1}}{(2i-1)!}$$

Primjeri izvršavanja programa.

```
Upisite x i n > 1.5 3↵
sinus(1.5000000000000000) = 1.0007812500000000↵
sin(1.5000000000000000) = 0.997494986604054↵
razlika = 0.003286263395946
```

```
Upisite x i n > 1.5 9↵
sinus(1.5000000000000000) = 0.997494986604073
sin(1.5000000000000000) = 0.997494986604054
razlika = 0.0000000000000018
```

12. Za sljedeći program nacrtati pojednostavljenu sliku stoga u svakom trenutku u kojem se tijekom izvršavanja programa njegov sadržaj promijeni (onako kako je prikazano na predavanjima).

```
#include <stdio.h>

int f(int a, int b) {
    int rez;
    rez = a + b;

    return rez;
}

int g(int a, int b, int c) {
    int rez1, rez2;
    rez1 = f(a, b);
    rez2 = f(b, c);

    return rez1 * rez2;
}

int main(void) {
    int x;
    x = g(2, 3, 4);

    return 0;
}
```

13. Napisati rekurzivnu funkciju naziva `fibonacci`. Matematički izraz kojim je definirana funkcija je:

$$fibonacci(n) = \begin{cases} 1 & \text{za } n = 1, 2 \\ fibonacci(n-1) + fibonacci(n-2) & \text{za } n > 2 \end{cases}$$

U glavnom programu s tipkovnice učitati cijeli broj n koji predstavlja redni broj člana niza, pozivom funkcije izračunati, a zatim ispisati n -ti član niza.

Primjeri izvršavanja programa.

```
Upisite redni broj clana niza > 1↵  
fibonacci(1) = 1
```

```
Upisite redni broj clana niza > 40↵  
fibonacci(40) = 102334155
```

Rješenja:

1. `#include <stdio.h>`

```
int kvadrat(int n) {
    int kv;
    kv = n * n;
    return kv;
}

int main(void) {
    int arg, rez;
    printf("Upisite cijeli broj > ");
    scanf("%d", &arg);
    rez = kvadrat(arg);
    printf("%d na kvadrat jest %d", arg, rez);

    return 0;
}
```

2. 50000^2 prelazi dopušteni raspon za tip podatka int. Funkcija će zato vratiti neispravan rezultat -1794967296. Potrebno je povećati raspon brojeva kojeg dopušta tip funkcije, varijable koja se u funkciji koristi za izračunavanje tog broja, te u izrazu kojim se izračunava kvadrat osigurati da se operacija obavlja u domeni s povećanim rasponom i povećati raspon vrijednosti varijable koja se koristi za pohranu rezultata poziva funkcije.

```
#include <stdio.h>

unsigned long long kvadrat(int n) {
    unsigned long long kv;
    kv = (unsigned long long)n * n;
    return kv;
}

int main(void) {
    int arg;
    unsigned long long rez;
    printf("Upisite cijeli broj > ");
    scanf("%d", &arg);
    rez = kvadrat(arg);
    printf("%d na kvadrat jest %llu", arg, rez);

    return 0;
}
```

3. Vrijednost argumenta 3.5 implicitno će se pretvoriti u cijeli broj 3 jer je parametar cijeli broj. Izračunat će se cijeli broj 9 i vratiti u glavni program. Zato je potrebno tip podatka `int` zamijeniti tipom podatka `float` (ili `double`), kao što je prikazano u nastavku.

```
#include <stdio.h>

float kvadrat(float n) {
    float kv;
    kv = n * n;
    return kv;
}

int main(void) {
    float arg;
    float rez;
    printf("Upisite realni broj > ");
    scanf("%f", &arg);
    rez = kvadrat(arg);
    printf("%f na kvadrat jest %f", arg, rez);

    return 0;
}
```

4. `#include <stdbool.h>`
`#include <stdio.h>`
- ```
bool jestVelikoSlovo(char c) {
 bool jestVeliko;
 jestVeliko = c >= 'A' && c <= 'Z';
 return jestVeliko;
}

int main(void) {
 char znak;
 bool rez;
 printf("Upisite znak > ");
 scanf("%c", &znak);

 rez = jestVelikoSlovo(znak);
 if (rez) {
 printf("Jest veliko slovo");
 } else {
 printf("Nije veliko slovo");
 }

 return 0;
}
```

ili

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

bool jestVelikoSlovo(char c) {
 return c >= 'A' && c <= 'Z';
}

int main(void) {
 char znak;
 printf("Upisite znak > ");
 scanf("%c", &znak);

 if (jestVelikoSlovo(znak)) {
 printf("Jest veliko slovo");
 } else {
 printf("Nije veliko slovo");
 }

 return 0;
}
```

## 5. #include &lt;stdio.h&gt;

```
void tablica(int redaka, int stupaca) {
 int i, j;
 // ispisi prvi red: "zaglavlje" tablice
 printf(" ");
 for (j = 1; j <= stupaca; ++j)
 printf("%5d", j);
 printf("\n");

 // ispisi tablicu
 for (i = 1; i <= redaka; ++i) {
 // na pocetku svakog retka ispisi redni broj retka
 printf("%5d", i);

 for (j = 1; j <= stupaca; ++j) {
 printf("%5d", i * j);
 }
 printf("\n");
 }
}

int main(void) {
 int m, n;
 printf("Upisite broj redaka i stupaca > ");
 scanf("%d %d", &m, &n);
 tablica(m, n);

 return 0;
}
```

## 6. #include &lt;stdio.h&gt;

```
#define MAXCLAN 15

void fibonacci15(void) {
 int i;
 int fbroj[MAXCLAN];

 fbroj[0] = fbroj[1] = 1;
 for (i = 2; i < MAXCLAN; ++i) {
 fbroj[i] = fbroj[i - 1] + fbroj[i - 2];
 }

 for (i = 0; i < MAXCLAN; ++i) {
 if (i > 0)
 printf(", ");
 printf("%d", fbroj[i]);
 }

 return;
}

int main(void) {
 fibonacci15();

 return 0;
}
```

## 7. #include &lt;stdio.h&gt;

```
void fibonacciN(int n) {
 int i;
 int fbroj[n];

 if (n > 0)
 fbroj[0] = 1;
 if (n > 1)
 fbroj[1] = 1;
 for (i = 2; i < n; ++i) {
 fbroj[i] = fbroj[i - 1] + fbroj[i - 2];
 }
 for (i = 0; i < n; ++i) {
 if (i > 0)
 printf(", ");
 printf("%d", fbroj[i]);
 }

 return;
}

int main(void) {
 int n;
 printf("Upisite broj clanova > ");
 scanf("%d", &n);
 fibonacciN(n);

 return 0;
}
```



8. #include <stdio.h>

```
int kolikiJeInt(void) {
 return sizeof(int);
}

int main(void) {
 printf("Na ovom racunalu int koristi bajtova: %d", kolikiJeInt());

 return 0;
}
```

9. #include <stdio.h>

```
int getBit(unsigned int x, unsigned int n) {
 return x >> n & 0x1;
}

int main(void) {
 unsigned int broj, n;

 printf("Upisite nenegativni cijeli broj > ");
 scanf("%u", &broj);
 printf("Upisite redni broj bita > ");
 scanf("%u", &n);

 printf("Vrijednost bita je %d ", getBit(broj, n));

 return 0;
}
```

10. #include <stdio.h>

```
int getBit(unsigned int x, unsigned int n) {
 return x >> n & 0x1;
}

void printBinary(unsigned int x) {
 int i;
 for (i = 31; i >= 0; --i) {
 printf("%d", getBit(x, i));
 }

 return;
}

int main(void) {
 unsigned int broj;

 printf("Upisite nenegativni cijeli broj > ");
 scanf("%u", &broj);

 printBinary(broj);

 return 0;
}
```

```

11. #include <math.h>
 #include <stdio.h>

 unsigned long long fakt(unsigned int n) {
 unsigned int i;
 unsigned long long umnozak = 1ULL;
 for (i = 2U; i <= n; ++i)
 umnozak = umnozak * i;
 return umnozak;
 }

 double sinus(double x, unsigned int n) {
 unsigned int i;
 int predznak = 1;
 double clan, suma = 0.;
 for (i = 1U; i <= n; ++i) {
 clan = predznak * pow(x, 2 * i - 1) / fakt(2 * i - 1);
 suma = suma + clan;
 // priprema predznaka za sljedeci korak
 predznak = -predznak;
 }
 return suma;
 }

 int main(void) {
 double x;
 unsigned int n;
 double rezSin, rezSinus;

 printf("Upisite x i n > ");
 scanf("%lf %u", &x, &n);

 rezSinus = sinus(x, n);
 rezSin = sin(x);

 printf("sinus(%.15lf) = %.15lf\n", x, rezSinus);
 printf(" sin(%.15lf) = %.15lf\n", x, rezSin);
 printf(" razlika = %.15lf", rezSinus - rezSin);
 return 0;
 }

```

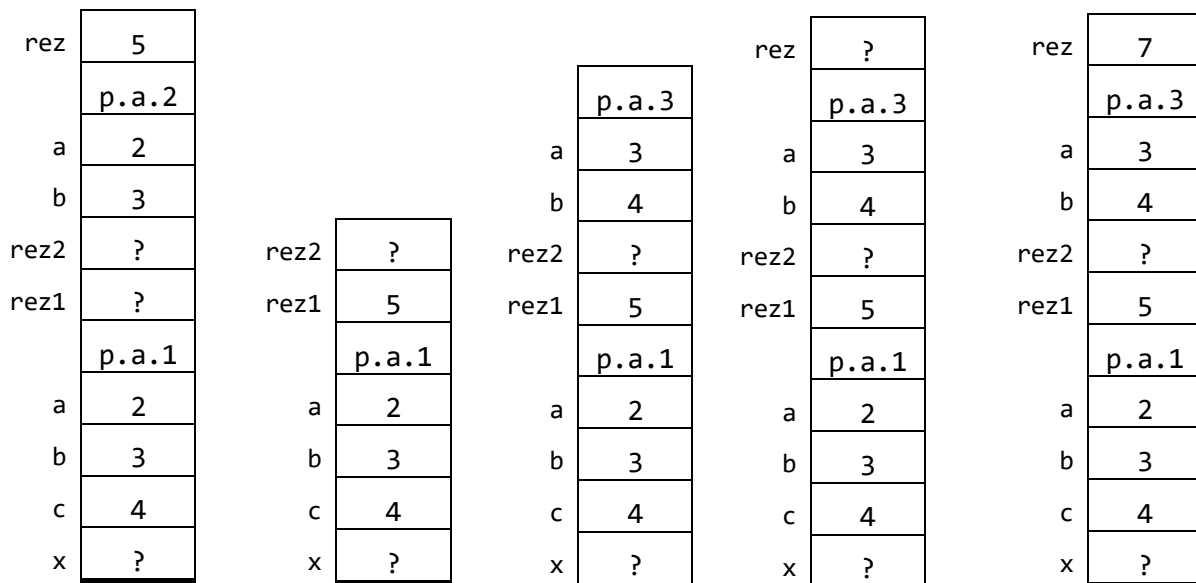
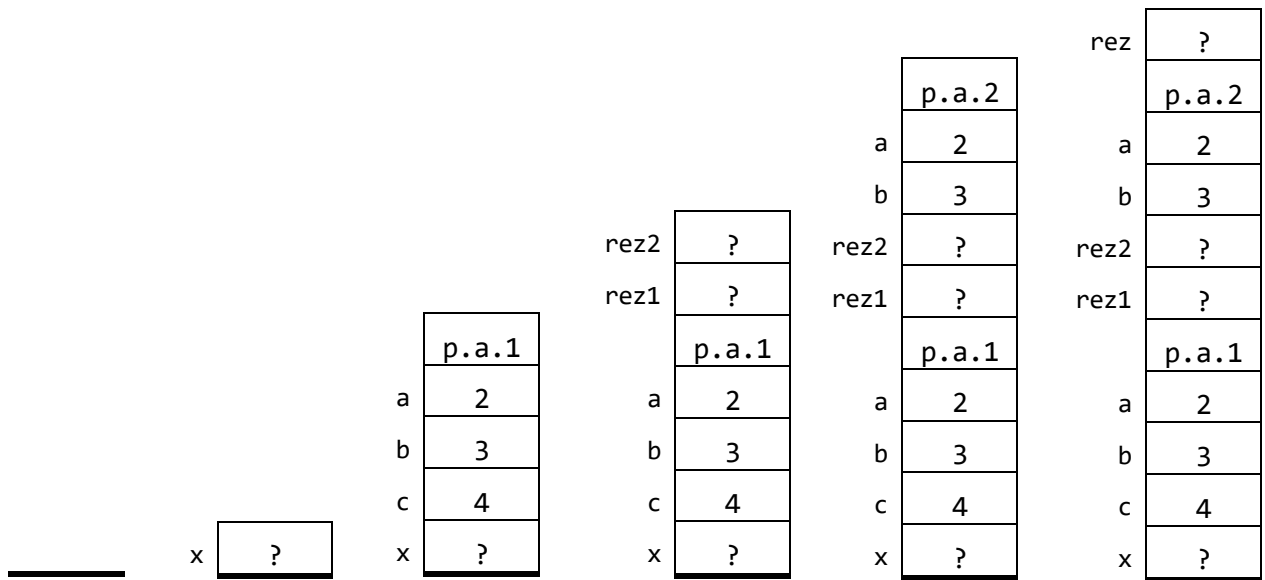
ili funkcija sinus bez zasebno napisane funkcije fakt: fakt(i+2) se izračunava iz fakt(i) iz prethodnog koraka.

```

double sinus(double x, unsigned int n) {
 unsigned int i;
 int predznak = 1;
 double clan;
 double suma = 0., xPot = x;
 unsigned long long fakt = 1ULL;
 for (i = 1; i <= n; ++i) {
 clan = predznak * xPot / fakt;
 suma = suma + clan;
 // priprema za sljedeci korak
 xPot = xPot * x * x;
 fakt = fakt * (2 * i) * (2 * i + 1);
 predznak = -predznak;
 }
 return suma;
}

```

12.



|      |       |   |    |
|------|-------|---|----|
| rez2 | 7     |   |    |
| rez1 | 5     |   |    |
|      | p.a.1 |   |    |
| a    | 2     |   |    |
| b    | 3     |   |    |
| c    | 4     |   |    |
| x    | ?     | x | 35 |

---

13. #include <stdio.h>

```
int fibonacci(int n) {
 int rez;
 if (n == 1 || n == 2) {
 rez = 1;
 } else {
 rez = fibonacci(n - 1) + fibonacci(n - 2);
 }
 return rez;
}
```

```
int main(void) {
 int n, rez;
 printf("Upisite redni broj clana niza > ");
 scanf("%d", &n);
 rez = fibonacci(n);
 printf("fibonacci(%d) = %d", n, rez);

 return 0;
}
```

ili

```
int fibonacci(int n) {
 if (n == 1 || n == 2) {
 return 1;
 } else {
 return fibonacci(n - 1) + fibonacci(n - 2);
 }
}
```

```
int main(void) {
 int n;
 printf("Upisite redni broj clana niza > ");
 scanf("%d", &n);
 printf("fibonacci(%d) = %d", n, fibonacci(n));

 return 0;
}
```