Природно-математички факултет
Универзитет у Бањој Луци

Функције и рекурзије - Група 3 и 4

Аутори: Драган Матић, Милана Грбић,
Милан Предојевић, Ненад Вилендечић

2025

Садржај

Понављање

1. Задатак: Исцртавање

Написати програм који за унесени непаран позитиван број n коришћењем знака * исцртава велико плус димензије n. Напомена: Претпоставити да је унос исправан.

Примјер извршавања

```
Unesite neparan pozitivan broj n: 5
  *
  *
*****
  *
  *
```

Рјешење

```cpp
#include <iostream>
using namespace std;

int main() {
    int n;
    cout << "Unesite neparan pozitivan broj n: ";
    cin >> n;

    // Iscrtavanje plus znaka
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i == n / 2 || j == n / 2) {
                //Ispisivanje * na srednjem redu i srednjoj koloni
                cout << "*";
            } else {
                cout << " ";  // Prazno polje za ostale pozicije
            }
        }
        cout << endl;  // Prelazak u novi red
    }

    return 0;
}
```

Увод

Функције су блокови кода који обављају одређени задатак и могу бити позвани више пута током извршавања програма. У C++-у, функције могу примати аргументе (параметре) и враћати резултат (повратну вриједност).

Дефиниција функције

Функција у C++-у се дефинише са сљедећом синтаксом:

```
tip_povratne_vrijednosti ime_funkcije(tip parametar1, tip parametar2, ...) {
    // tijelo funkcije
    return vrijednost;
}
```

- **tip_povratne_vrijednosti** је тип података који функција враћа (нпр. `int`, `double`, `void` ако функција не враћа ништа).
- **ime_funkcije** је назив који користимо приликом дефинисања и позивања функције.
- **parametri** су вриједности које функција прима.

На примјер, функција која сабира два броја може изгледати овако:

```
int saberi(int a, int b) {
    return a + b;
}
```

Ова функција прима два параметра типа `int` и враћа њихов збир као `int`.

Позивање функције

Након што је функција дефинисана, може се позвати из главног дијела програма или из других функција:

```
#include <iostream>
using namespace std;

int saberi(int a, int b) {
    return a + b;
}

int main() {
    int rezultat = saberi(5, 7);  // Pozivamo funkcije saberi
    cout << "Rezultat sabiranja: " << rezultat << endl;
    return 0;
}
```

У овом примјеру, функција `saberi` се позива са аргументима 5 и 7, а резултат се чува у промјенљивој `rezultat`.

Функције које не враћају вриједност

Ако функција не треба да врати резултат, користи се тип `void`. Овакве функције обично служе за извршавање неких акција, али не враћају вриједности. На примјер:

```
void ispisiPoruku() {
    cout << "Dobro dosli na skolu programiranja :)" << endl;
}
```

У овом случају, функција `ispisiPoruku` не прима параметре и не враћа никакву вредност. Када се позове, само исписује поруку на екрану.

Пренос параметара

Параметри функција могу се преносити на два начина:

- **по вриједности** - функција добија копију вредности параметра.
- **по референци** - функција добија референцу на оригинални параметар, што значи да може мијењати његову вриједност.

Примјер преноса по вриједности:

```
void povecaj(int broj) {
    broj = broj + 1;
}

int main() {
    int x = 5;
    povecaj(x);  // x ostaje 5
    cout << x << endl;  // 5
}
```

Примјер преноса по референци:

```
void povecaj(int &broj) {
    broj = broj + 1;
}

int main() {
    int x = 5;
    povecaj(x);  // x ce biti 6
    cout << x << endl;  // 6
}
```

Пренос више параметара

Функције у C++-у могу примити било који број параметара. На примјер:

```
int proizvod(int a, int b, int c) {
    return a * b * c;
}
```

Ова функција прима три параметра и враћа њихов производ.

Рекурзивне функције

Рекурзивна функција је функција која позива саму себе како би ријешила мањи потпроблем оригиналног проблема.

Примјер рекурзивне функције за израчунавање факторијела:

```
#include <iostream>
using namespace std;

int faktorijel(int n) {
    if (n == 0) return 1;
    return n * faktorijel(n - 1);
}

int main() {
    int broj;
    cout << "Unesite broj: ";
    cin >> broj;
    cout << "Faktorijel broja " << broj << " je " <<
   faktorijel(broj) << endl;
    return 0;
}
```

Објашњење:

- Основни случај: Ако је `n == 0`, функција враћа 1.
- Рекурзивни случај: Функција рачуна `n * faktorijel(n - 1)`.
- Позиви се понављају све док се не достигне основни случај.

Задаци

1. Задатак: Минимални број

Написати функцију која прима три броја и враћа најмањи од њих.

Примјер извршавања

```
Unesite tri broja: 5 2 8
Najmanji broj je: 2
```

Рјешење

```cpp
#include <iostream>
using namespace std;

int najmanji(int a, int b, int c) {
    if (a<=b && a<=c)
        return a;

    if (b<=a && b<=c)
        return b;

    retun c;
}

int main() {
    int x, y, z;
    cout << "Unesite tri broja: ";
    cin >> x >> y >> z;
    cout << "Najmanji broj je: " << najmanji(x, y, z) << endl;
    return 0;
}
```

2. Задатак: Прост број

Написати функцију која провјерава да ли је број прост.

Примјер извршавања

```
Unesite broj: 7
Broj je prost.
```

Рјешење

```cpp
#include <iostream>
using namespace std;

bool jeProst(int n) {
    if (n < 2) return false;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return false;
    }
    return true;
}

int main() {
    int n;
    cout << "Unesite broj: ";
    cin >> n;
    if (jeProst(n))
        cout << "Broj je prost." << endl;
    else
        cout << "Broj nije prost." << endl;
    return 0;
}
```

3. Задатак: Највећи заједнички делилац

Написати функцију која рачуна НЗД два броја.

Примјер извршавања

```
Unesite dva broja: 24 18
NZD je: 6
```

Рјешење

```cpp
#include <iostream>
using namespace std;

int nzd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int main() {
    int x, y;
    cout << "Unesite dva broja: ";
    cin >> x >> y;
    cout << "NZD je: " << nzd(x, y) << endl;
    return 0;
}
```

4. Задатак: Бројање цифара

Написати функцију која броји број цифара у датом броју.

Примјер извршавања

```
Unesite broj: 12345
Broj ima 5 cifara.
```

Рјешење

```cpp
#include <iostream>
using namespace std;

int brojCifara(int n) {
    int brojac = 0;
    while (n) {
        brojac++;
        n /= 10;
    }
    return brojac;
}

int main() {
    int n;
    cout << "Unesite broj: ";
    cin >> n;
    cout << "Broj ima " << brojCifara(n) << " cifara." << endl;
    return 0;
}
```

5. Задатак: n-ти Фибоначијев број

Написати рекурзивну функцију и итеративну функцију које рачунају n-ти Фибоначијев број. Функције тестирати у главном дијелу програма на броју n који се уноси са тастатуре.

Примјер извршавања

```
Unesite broj n: 5
Fibonacci broj (rekurzivno) za n = 5 je: 8
Fibonacci broj (iterativno) za n = 5 je: 8
```

Рјешење

```cpp
#include <iostream>
using namespace std;

// Rekurzivna funkcija za racunanje n-tog Fibonacci broja
long long fib_rekurzivno(int n) {
    if (n == 0 || n == 1)
    return 1;
    else
    return fib_rekurzivno(n - 1) + fib_rekurzivno(n - 2);

}

// Iterativna funkcija za racunanje n-tog Fibonacci broja
long long fib_iterativno(int n) {
    if (n == 0 || n == 1)
    return 1;

    long long prvi = 1, drugi = 1, novi;
    for (int i = 2; i <= n; i++) {
        novi = prvi + drugi;
        prvi = drugi;
        drugi = novi;
    }
    return drugi;
}

int main() {
    int n;
    cout << "Unesite broj n: ";
    cin >> n;

    // Testiranje rekurzivne funkcije
    long long rezultat_rekurzivno = fib_rekurzivno(n);
    cout << "Fibonacci broj (rekurzivno) za n = " << n << " je: "
   << rezultat_rekurzivno << endl;

    // Testiranje iterativne funkcije
```

```
37      long long rezultat_iterativno = fib_iterativno(n);
38      cout << "Fibonacci broj (iterativno) za n = " << n << " je: "
    << rezultat_iterativno << endl;
39
40      return 0;
41  }
```

Домаћи рад

1. Задатак: Пар непар

Написати функцију која као параметар узима цио број и провјерава да ли су његове цифре наизмјенично парне и непарне. Функција треба да врати *true* ако цифре испуњавају услов, а *false* иначе.

Написати програм који од корисника захтијева унос позитивног цијелог броја n, а затим на случајан начин генерише n бројева из интервала $[-10000, -100] \cup [100, 10000]$, за које треба испитати да ли испуњавају поменути услов.

Примјер извршавања

```
Unesite broj n: 4
Generisan: -478
Broj -487 ispunjava uslov
Generisan: 2574
Broj 2574 ne ispunjava uslov
Generisan: 3492
Broj 3492 ispunjava uslov
Generisan: 285
Broj 285 ne ispunjava uslov
```

2. Задатак: Ротација

Написати функцију *void* rotacija(*int* &n) која ротира цифре броја за једну позицију улијево, односно прву цифру премјешта на крај.

Написати програм који за бројеве који се уносе све док се не унесе број нула исписује одговарајуће ротације броја.

Примјер извршавања

```
Unesite broj: 146
Novi broj: 461
Unesite broj: 18
Novi broj: 81
Unesite broj: 3856
Novi broj: 8563
Unesite broj: 7
Novi broj: 7
Unesite broj: 0
```