# Zadatak 1.

Napravite program u kome ćete računati rješenja kvadratne jednačine *ax2 + bx + c = 0*: pomoću funkcije čiji prototip (deklaracija) glasi:

void f1(float a, float b, float c, float& r1, float& r2, bool& greska);

Formalni parametri r1 i r2 predstavljaju **r**ješenja kvadratna jednačine.

Formalni parametar greska treba da ima vrijednost 0 (false) ako je bilo moguće izračunati rješenje. U slučaju da nije moguće izračunati rješenje, vrijednost varijable greska treba da bude 1 (true).

Formule za računanje rješenja kvadratne jednačine glase:

, gdje je , uslovi su:  i 

*Pomoć:*

1. u main-u vršite unos parametara a, b i c
2. u main-u deklarišite varijablu error, x1, x2
3. iz main-a pozovite funkciju f1 i proslijedite joj aktuelne parametre a, b, c, x1, x2, error
   * cilj funkcije f1 je da na osnovu vrijednosti a, b i c izračuna rješenja kvadratne jednačine i smjesti ih u varijable x1 i x2
   * ako je moguće izračunati rješenja (x1, x2) varijabla error nakon poziva funkcije treba da bude jednaka 0 (false), a ako nije bilo moguće izračunati varijabla error treba da bude jednaka 1 (true)
4. ispišite rješenja kvadratne jednačine samo u slučaju da postoje rješenja

|  |
| --- |
| **Rješenje** |
| #include <iostream>  #include <math.h>  using namespace std;  void f1(float a, float b, float c, float &r1, float &r2, bool& greska)  {  float D;  D = b\*b - 4 \* a\*c;  if (D<0 || a == 0)  greska = true; //ili greska = 1;  else  {  r1 = (-b + sqrt(D)) / (2 \* a);  r2 = (-b - sqrt(D)) / (2 \* a);  greska = false; //ili greska = 0;  }  }  void main()  {  float a, b, c, x1, x2;  bool error;  cout << "Unesite a, b, c \n";  cin >> a >> b >> c;  f1(a, b, c, x1, x2, error);  if (error) //cita se: "ako je (greska) onda..."  cout << "Greska, kvadratna jednacina nema rjesenja \n";  else  {  cout << "x1 = " << x1 << endl;  cout << "x2 = " << x2 << endl;  }  } |

# Zadatak 2.

Prepravite zadatak 1 tako da koristite pokazivač pf1 na funkciju f1.

Poziv funkcije vršite pomoću pokazivača pf1.

|  |
| --- |
| **Rješenje** |
| #include <iostream>  #include <math.h>  using namespace std;  void f1(float a, float b, float c, float &r1, float &r2, bool& greska)  {  float(\*pk)(float) = sqrt;  float D;  D = b\*b - 4 \* a\*c;  if (D<0 || a == 0)  greska = true; //ili greska = 1;  else  {  r1 = (-b + pk(D)) / (2 \* a);  r2 = (-b - pk(D)) / (2 \* a);  greska = false; //ili greska = 0;  }  }  void main()  {  void(\*pok1)(float, float, float, float &, float &, bool&) = f1;  float a, b, c, x1, x2;  bool error;  cout << "Unesite a, b, c \n";  cin >> a >> b >> c;  pok1(a, b, c, x1, x2, error);  if (error) //cita se: "ako je (greska) onda..."  cout << "Greska, kvadratna jednacina nema rjesenja \n";  else  {  cout << "x1 = " << x1 << endl;  cout << "x2 = " << x2 << endl;  }  } |

# Zadatak 3.

Prethodni zadatak riješite pomoću funkcije

void f1(float a, float b, float c, float\* r1, float\* r2, bool\* greska);

# Zadatak 4.

Prepravite zadatak 3 tako da koristite pokazivač pf2 na funkciju f2.

Poziv funkcije vršite pomoću pokazivača pf2.

|  |
| --- |
| **Rješenje** |
| #include <iostream>  #include <math.h>  using namespace std;  void f2(float a, float b, float c, float\* r1, float\* r2, bool\* greska) {  float(\*pk)(float) = sqrt;  float D;  D = b\*b - 4 \* a\*c;  if (D<0 || a == 0)  \*greska = true; //ili greska = 1;  else  {  \*r1 = (-b + pk(D)) / (2 \* a);  \*r2 = (-b - pk(D)) / (2 \* a);  \*greska = false; //ili greska = 0;  }  }  void main()  {  void(\*pok2)(float, float, float, float\*, float\*, bool\*) = f2;  float a, b, c, x1, x2;  bool error;  cout << "Unesite a, b, c \n";  cin >> a >> b >> c;  pok2(a, b, c, &x1, &x2, &error);  if (error) //cita se: "ako je (greska) onda..."  cout << "Greska, kvadratna jednacina nema rjesenja \n";  else  {  cout << "x1 = " << x1 << endl;  cout << "x2 = " << x2 << endl;  }  } |

# Zadatak 5 – Pokazivač na funkciju

Dovršite funkcije;

|  |
| --- |
| #include <iostream>  using namespace std;  typedef void (FunkTip)();  void zadatak1()  {  //pauziranje na 1 sek  \_sleep(1000);  cout << "radni zadatak 1 zavrsen" << endl;  }  void zadatak2()  {  //pauziranje na 1 sek  \_sleep(1000);  cout << "radni zadatak 2 zavrsen" << endl;  }  void zadatak3()  {  //pauziranje na 1 sek  \_sleep(1000);  cout << "radni zadatak 3 zavrsen" << endl;  }  void zadatak4()  {  //pauziranje na 1 sek  \_sleep(1000);  cout << "radni zadatak 4 zavrsen" << endl;  }  struct TodoRaspored  {  int brojacZadatakaNiz = 0;  FunkTip\* zadaci[10];  };  void dodaj\_zadatak(TodoRaspored\* todo, FunkTip\* newZadatak)  {  //implementirati: u niz **zadaci** unutar **todo** dodati **newZadatak**  }  void ukloni\_zadatak(TodoRaspored\* todo)  {  //implementirati:  //a) učitati redni broj **x**,  //b) za pozicije **x** iz niza **todo->zadaci** ukloniti elemenat  // (elemenat sa zadnje pozicije premjestiti/kopirati na poziciju **x**)  }  void izvrsi\_zadatke(TodoRaspored\* todo)  {  //implementirati:  //za svaki pokazivač **p** iz niza **todo->zadaci** uradi sljedeće  // - pozovi funkciju preko pokazivača na funkciju p.  }  void main()  {  TodoRaspored\* danas = new TodoRaspored;  cout << "TodoRaspored danas" << endl;  dodaj\_zadatak(danas, zadatak1);  dodaj\_zadatak(danas, zadatak2);  FunkTip\* pZ3 = zadatak3;  dodaj\_zadatak(danas, pZ3);  TodoRaspored\* sutra = new TodoRaspored;  cout << "TodoRaspored sutra" << endl;  dodaj\_zadatak(sutra, zadatak2);  dodaj\_zadatak(sutra, zadatak3);  dodaj\_zadatak(sutra, zadatak4);  izvrsi\_zadatke(danas);  system("pause");  izvrsi\_zadatke(sutra);  } |

|  |
| --- |
| Rješenje |
| void dodaj\_zadatak(TodoRaspored\* todo, FunkTip\* newZadatak)  {  todo->zadaci[todo->brojacZadatakaNiz] = newZadatak;  todo->brojacZadatakaNiz++;  }  void ukloni\_zadatak(TodoRaspored\* todo)  {  int rb;  cin >> rb;  todo->brojacZadatakaNiz--;  todo->zadaci[rb] = todo->zadaci[todo->brojacZadatakaNiz];  }  void izvrsi\_zadatke(TodoRaspored\* todo)  {  for (int i = 0; i < todo->brojacZadatakaNiz; i++)  {  FunkTip\* p = todo->zadaci[i];  p();  }  } |

# Zadatak 6 – „Pokazivač na funkciju“ kao povratna vrijednost funkcije

1. U prethodni program dodajte sljedeću funkciju

|  |
| --- |
| FunkTip\* odaberi\_zadatak()  {  cout << "Odaberite zadatak 1 - 4: " << endl;  int r;  cin >> r;  if (r == 1)  return zadatak1;  if (r == 2)  return zadatak2;  if (r == 3)  return zadatak2;  if (r == 4)  return zadatak2;  return nullptr;  } |

Prepravite funkciju main tako da koristite funkciju odaberi\_zadatak;

|  |
| --- |
| **Rješenje** |
| #include <iostream>  using namespace std;  typedef void (FunkTip)();  void zadatak1()  {  //pauziranje na 1 sek  \_sleep(1000);  cout << "radni zadatak 1 zavrsen" << endl;  }  void zadatak2()  {  //pauziranje na 1 sek  \_sleep(1000);  cout << "radni zadatak 2 zavrsen" << endl;  }  void zadatak3()  {  //pauziranje na 1 sek  \_sleep(1000);  cout << "radni zadatak 3 zavrsen" << endl;  }  void zadatak4()  {  //pauziranje na 1 sek  \_sleep(1000);  cout << "radni zadatak 4 zavrsen" << endl;  }  FunkTip\* odaberi\_zadatak()  {  cout << "Odaberite zadatak 1 - 4: " << endl;  int r;  cin >> r;  if (r == 1)  return zadatak1;  if (r == 2)  return zadatak2;  if (r == 3)  return zadatak2;  if (r == 4)  return zadatak2;  return nullptr;  }  struct TodoRaspored  {  int brojacZadatakaNiz = 0;  FunkTip\* zadaci[10];  };  void dodaj\_zadatak(TodoRaspored\* todo, FunkTip\* newZadatak)  {  todo->zadaci[todo->brojacZadatakaNiz] = newZadatak;  todo->brojacZadatakaNiz++;  }  void ukloni\_zadatak(TodoRaspored\* todo)  {  int rb;  cin >> rb;  todo->brojacZadatakaNiz--;  todo->zadaci[rb] = todo->zadaci[todo->brojacZadatakaNiz];  }  void izvrsi\_zadatke(TodoRaspored\* todo)  {  for (int i = 0; i < todo->brojacZadatakaNiz; i++)  {  FunkTip\* p = todo->zadaci[i];  p();  }  }  void main()  {  TodoRaspored\* danas = new TodoRaspored;  cout << "TodoRaspored danas" << endl;  dodaj\_zadatak(danas, odaberi\_zadatak());  dodaj\_zadatak(danas, odaberi\_zadatak());  FunkTip\* pZ3 = odaberi\_zadatak();  dodaj\_zadatak(danas, pZ3);  TodoRaspored\* sutra = new TodoRaspored;  cout << "TodoRaspored sutra" << endl;  dodaj\_zadatak(sutra, odaberi\_zadatak());  dodaj\_zadatak(sutra, odaberi\_zadatak());  dodaj\_zadatak(sutra, odaberi\_zadatak());  izvrsi\_zadatke(danas);  system("pause");  izvrsi\_zadatke(sutra);  } |