Umetnost programiranja v C++

Borko Bošković, Blaž Pšeničnik

21. avgust 2026

Uvod

Spoznali bomo

1. Koncepti programskega jezika C++
2. Razvojno orode
3. Veščine in znanja iz programiranja
4. Izdlava računalniških iger
5. Obdelava slik

Termini delavnice (učilnica: G2-N3 Seminarska soba)

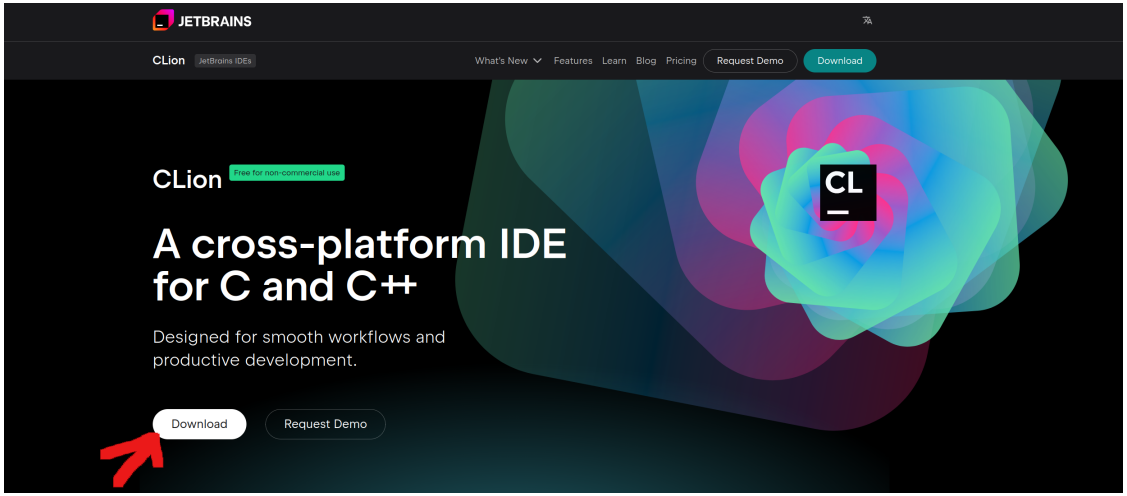
1. 19. 8. 2026 od 9:00 do 15:00
2. 20. 8. 2026 od 9:00 do 14:15
3. 21. 8. 2026 od 9:00 do 14:15

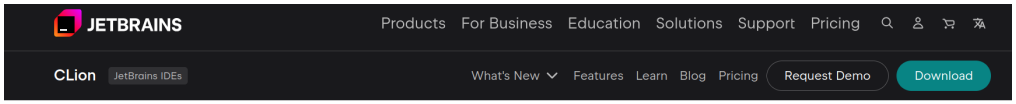
Dokumenti

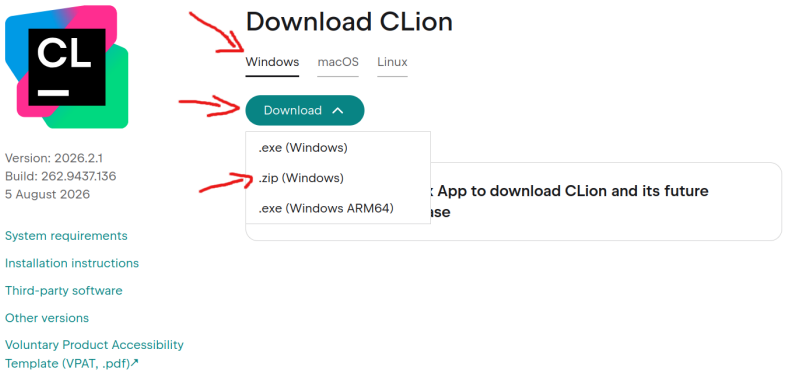
1. Izpolnjeno in podpisano potrdilo s strani staršev oz. skrbnikov.
2. Seznam prisotnosti
3. Boni za malico in sladoled
4. Prijava: dijak / dijak2026
5. Preverjanje znanja: `https://tinyurl.com/umetnostcpp`
6. Anketa o zadovoljstvi:
   `https://docs.google.com/forms/d/e/1FAIpQLSd48nb9ZN4C2EYILn7-N5FBqdJTgKatLcxHf8I1Jx68WFF_Uw/viewform?usp=dialog`

Razvojno okolje

Razvojno okolje: CLion

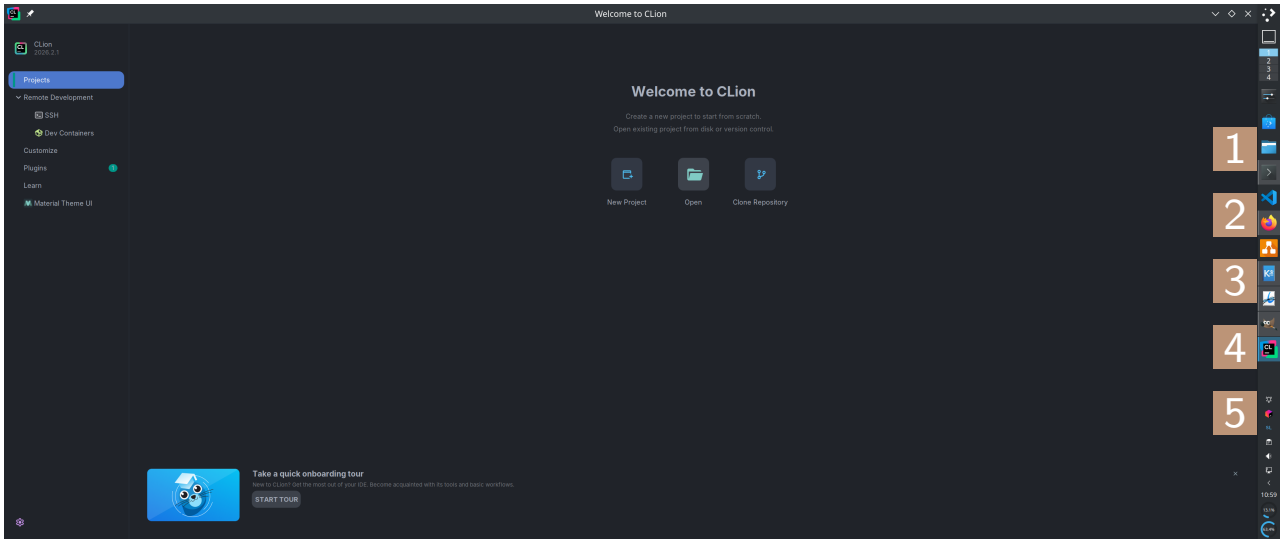






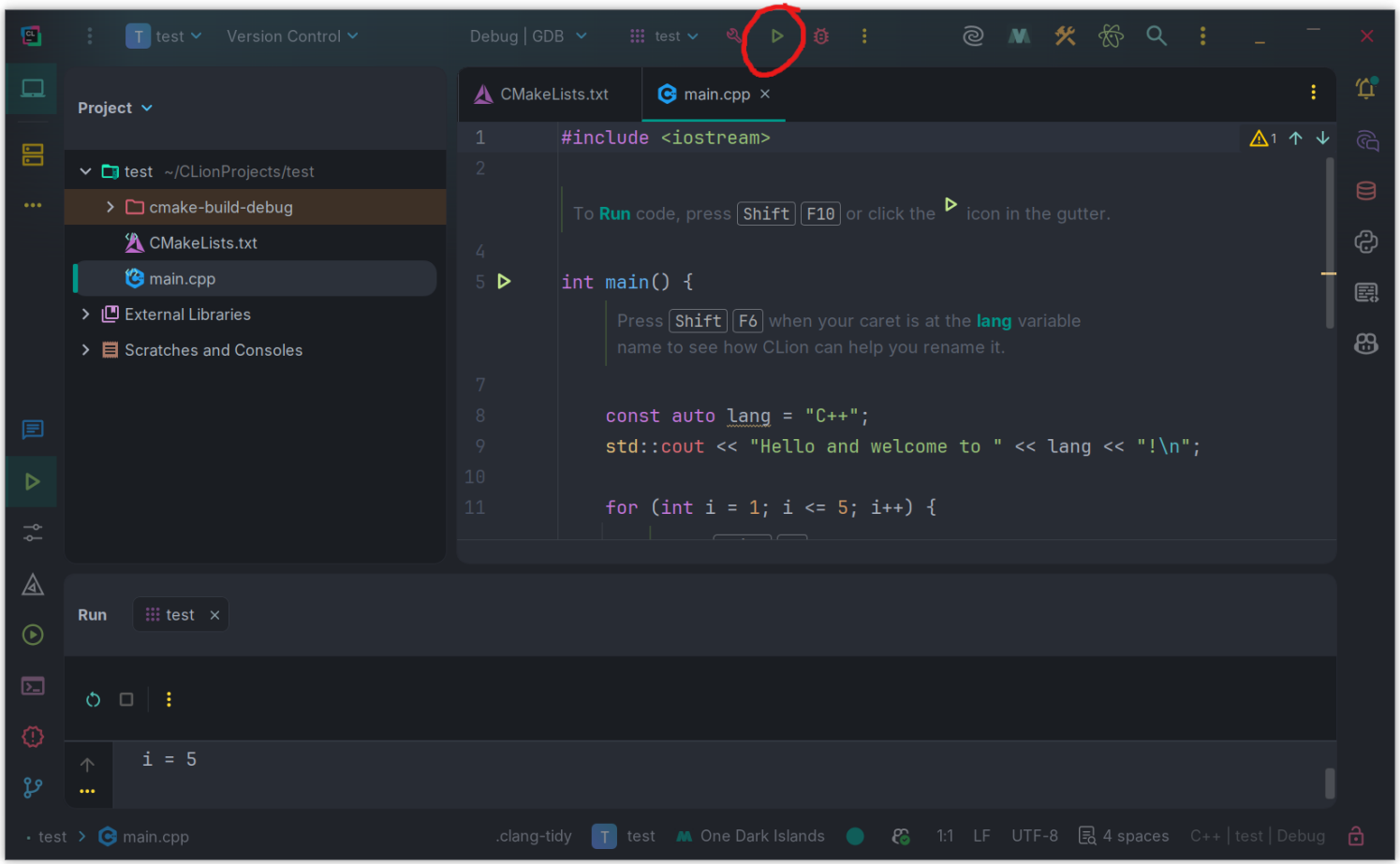
1. `https://www.jetbrains.com/clion/`
2. Download
3. Windows/Download/zip(Windows)

Ustvarjanje novega projekta



1. Odpri CLion
2. Klikni **New project**
3. Vpiši ime **Location: .../test**
4. Language standard: C++20
5. Klikni **Create**

Prevajanje in zagon



Osnove C++

Osnovni izpis

Program, ki izpiše tvoje ime in starost.

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

int main() {
      cout << "Moje ime je ..." << endl;
      cout << "Star sem ..." << endl;
      return 0;
}
```

Preprosti izračuni

Program, ki izračuna obseg in ploščino pravokotnika.

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

int main() {
      double a, b;
      cout << "Vnesi stranici: ";
      cin >> a >> b;
      cout << "Obseg:" << 2*a+2*b <<endl;
      cout << "Ploscina:" << a*b <<endl;
      return 0;
}
```

Pogojni stavki

Program, ki preveri, ali je vpisano število sodo ali liho.

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

int main() {
      int stevilo;
      cout << "Vnesi stevilo: ";
      cin >> stevilo;
      if (stevilo % 2 == 0) {
         cout << "Sodo" << endl;
      } else {
         cout << "Liho" << endl;
      }
      return 0;
}
```

Program, ki preveri oceno

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

int main() {
    int ocena;
    cout << "Vnesi oceno (1-5): ";
    cin >> ocena;

    if(ocena == 5){
        cout << "Odlicno!";
    } else if(ocena == 4){
        cout << "Prav dobro!";
    }
    else if(ocena == 3){
        cout << "Dobro.";
    }
    else if(ocena == 2){
        cout << "Zadostno.";
    }
    else{
        cout << "Nezadostno.";
    }
    return 0;
}
```

Zanke v C++

- **for** – znano število ponovitev.
- **while** – ponavljanje dokler je pogoj resničen.
- **do-while** – vsaj enkrat izvedena.

```
for(int i=0; i<5; i++)
    cout << i << endl;

int x = 0;
while(x < 5){
    cout << x++ << endl;
}

int y = 0;
do {
    cout << y++ << endl;
} while(y < 5);
```

Izpis vseh sodih števil med 1 in 20

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

int main() {
    for(int i = 1; i <= 20; i++){
        if(i % 2 == 0){
            cout << i << " ";
        }
    }
    return 0;
}
```

Seštevanje števil do vnesenega n

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

int main() {
    int n;
    cout << "Vnesi n: ";
    cin >> n;
    int vsota = 0;
    int i = 1;
    while(i <= n){
        vsota += i;
        i++;
    }
    cout << "Vsota = " << vsota << endl;
    return 0;
}
```

Funkcije

- **Deklaracija**: pove prevajalniku ime, parametre in tip vrnjenega rezultata.
- **Definicija**: vsebuje dejansko kodo.

```
int vsota(int a, int b); // deklaracija

int main(){
    cout << vsota(3, 4); // klic
}

int vsota(int a, int b){ // definicija
    return a + b;
}
```

Funkcija, ki preveri, ali je število praštevilo

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

bool jePrastevilo(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 x;
    cin >> x;
    if(jePrastevilo(x))
        cout << "Prastevilo";
    else
        cout << "Ni prastevilo";
    return 0;
}
```

Prenos parametrov

- **Po vrednosti**: spremembe ne vplivajo na original.
- **Po referenci**: spremembe vplivajo na original.
- **S kazalcem**: spremembe vplivajo na original, uporablja se naslov.

```
void povecajVrednost(int x) { x++; }
void povecajReferenco(int &x) { x++; }
void povecajKazalec(int *x) { (*x)++; }

int main(){
    int a = 5;
    povecajVrednost(a);   // a = 5
    povecajReferenco(a); // a = 6
    povecajKazalec(&a);   // a = 7
    return 0;
}
```

Branje in pisanje v datoteke

- Knjižnica: `#include <fstream>`
- Razredi: `ofstream` (pisanje), `ifstream` (branje), `fstream` (oboje).

```
#include <fstream>
#include <string>
using namespace std;

int main(){
    // Pisanje v datoteko
    ofstream out("podatki.txt");
    out << "Pozdravljen svet!" << endl;
    out.close();

    // Branje iz datoteke
    ifstream in("podatki.txt");
    string vrstica;
    while(getline(in, vrstica)){
        cout << vrstica << endl;
    }
    in.close();
    return 0;
}
```

Program shrani seznam imen in jih nato prebere

```
#include <fstream>
#include <vector>
#include <string>
using namespace std;

int main(){
    // Pisanje
    ofstream out("imena.txt");
    out << "Ana\nBoris\nCene\n";
    out.close();

    // Branje
    ifstream in("imena.txt");
    string ime;
    while(getline(in, ime)){
        cout << "Prebrano ime: " << ime << endl;
    }
    in.close();
    return 0;
}
```

Naloga

Napiši program, ki:

1. Prebere 10 števil iz datoteke.
2. Izpiše največje in najmanjše število.
3. Vsa števila zapiše v drugo datoteko v obratnem vrstnem redu.
4. Za izračun najmanjšega in največjega števila uporabi funkcije.

Enodimenzionalna polja

- Polja omogočajo shranjevanje več vrednosti istega tipa.
- Dostop do elementov z indeksom od 0 naprej.

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

int main(){
    int stevila[5] = {1, 2, 3, 4, 5};
    for(int i = 0; i < 5; i++){
        cout << stevila[i] << " ";
    }
    return 0;
}
```

Dvodiemenzionalna polja

- Shranjevanje podatkov v obliki matrike.
- Indeksiranje: `polje[vrstica][stolpec]`.

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

int main(){
    int matrika[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
    for(int i = 0; i < 3; i++){
        for(int j = 0; j < 3; j++){
            cout << matrika[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Strukture

Strukture združujejo različne tipe podatkov v eno celoto.

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

struct Oseba {
    string ime;
    int starost;
};

int main(){
    Oseba o1;
    o1.ime = "Ana";
    o1.starost = 25;
    cout << o1.ime << ", " << o1.starost << " let" << endl;
    return 0;
}
```

Razredi – podobno kot strukture

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

class Avto {
public:
    string znamka;
    int leto;

    void izpisi(){
        cout << znamka << "␣(" << leto << ")" << endl;
    }
};
int main(){
    Avto a1;
    a1.znamka = "Toyota";
    a1.leto = 2020;
    a1.izpisi();
    return 0;
}
```

Dedovanje in polimorfizem – liki I

- Osnovni razred: `Telo` z virtualno metodo `ploscina()`.
- Podrazredi: `Kocka`, `Krogla`, `Valj`.
- V `main` shranimo kazalce na različne like v polje in seštejemo njihove površine.

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

class Telo { // Osnovni razred
public:
    virtual double ploscina() const = 0; // cista virtualna metoda
    virtual ~Telo() {} // destruktor
};

class Kocka : public Telo { // Podrazred: Kocka
    double a;
public:
    Kocka(double stranica) : a(stranica) {} // konstruktor
    double ploscina() const override { return 6 * a * a; }
};

class Krogla : public Telo { // Podrazred: Krogla
    double r;
```

Dedovanje in polimorfizem – liki II

```
public:
    Krogla(double radij) : r(radij) {}
    double ploscina() const override { return 4 * M_PI * r * r; }
};

// Podrazred: Valj
class Valj : public Telo {
    double r, h;
public:
    Valj(double radij, double visina) : r(radij), h(visina) {}
    double ploscina() const override { return 2 * M_PI * r * (r + h); }
};

int main() {
    Telo* liki[3];
    liki[0] = new Kocka(2);
    liki[1] = new Krogla(1);
    liki[2] = new Valj(1, 3);

    double vsota = 0;
    for(int i = 0; i < 3; i++) {
        vsota += liki[i]->ploscina();
        delete liki[i]; // sprostimo pomnilnik
    }
    cout << "Skupna povrsina: " << vsota << endl;
    return 0;
}
```

Naloga

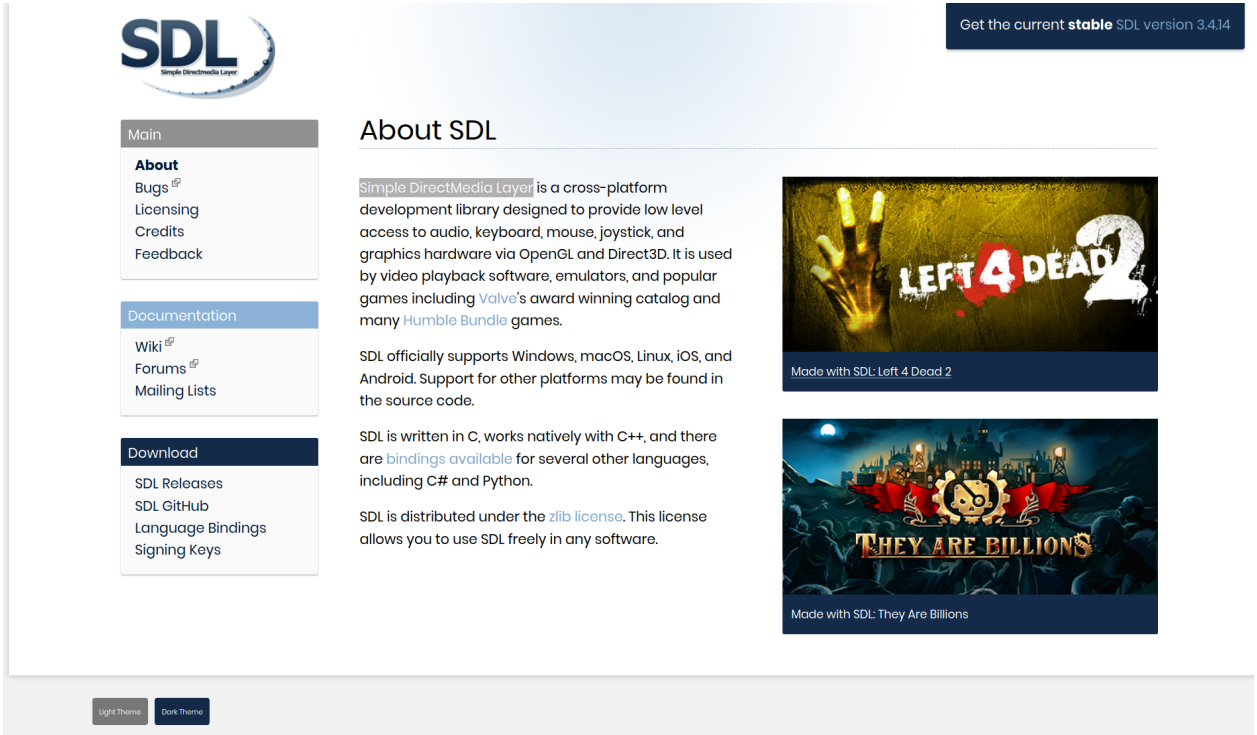
Napiši program, ki:

1. Ustvari polje 10 celih števil.
2. Izračuna povprečje vseh števil.
3. Uporabi strukturo za shranjevanje podatkov o študentu.
4. Uporabi razred za modeliranje bančnega računa z možnostjo pologa in dviga.

Simple DirectMedia Layer

Simple DirectMedia Layer (SDL)

https://www.libsdl.org/



Namestitev:

- CMD
- git clone `https://github.com/libsdl-org/SDL.git` vendored/SDL

Simple DirectMedia Layer (SDL)

SDL (Simple DirectMedia Layer) je knjižnica, ki omogoča delo z:

- okni,
- tipkovnico,
- miško,
- zvokom,
- grafiko,
- igralnimi ploščki.

Namestitev SDL

- CMD
- git clone `https://github.com/libsdl-org/SDL.git` vendored/SDL
- CMakefile

```
cmake_minimum_required(VERSION 4.3)
project(test)

set(CMAKE_CXX_STANDARD 20)

set(CMAKE_RUNTIME_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}")
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY "${CMAKE_BINARY_DIR}")

add_subdirectory(vendored/SDL EXCLUDE_FROM_ALL)
add_executable(test main.cpp)

target_link_libraries(test PRIVATE SDL3::SDL3)
```

Prvi program, ki uporablja knjižnico SDL3

Program:

- ustvariti okno,
- obdelovati dogodke s tipkovnice,
- risati na zaslon,
- premikati objekt,
- pravilno zaključiti program.

Končni rezultat bo preprost program, v katerem lahko s puščicami premikamo kvadrat.

Prvi program, ki uporablja knjižnico SDL3

```
#include <SDL3/SDL.h>

int main() {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window* window = SDL_CreateWindow("Moj prvi SDL program", 800, 600, 0);

    bool running = true;
    while (running) {
        SDL_Event event;
        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_EVENT_QUIT) { running = false; }
        }
    }

    SDL_DestroyWindow(window);
    SDL_Quit();
    return 0;
}
```

Dodamo grafiko

```
#include <SDL3/SDL.h>

int main() {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window* window;
    SDL_Renderer* renderer;
    SDL_CreateWindowAndRenderer( "Moj prvi SDL program", 800, 600, 0,
                                 &window, &renderer );

    bool running = true;
    while (running) {
        SDL_Event event;
        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_EVENT_QUIT) { running = false; }
        }

        SDL_SetRenderDrawColor( renderer, 30, 30, 30, 255 );
        SDL_RenderClear(renderer);
        SDL_RenderPresent(renderer);
    }

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();

    return 0;
}
```

Narišimo kvadrat

SDL omogoča risanje pravokotnikov. Najprej določimo njegovo velikost in položaj:

```
SDL_FRect square;

square.x = 350;
square.y = 250;
square.w = 100;
square.h = 100;
```

Nato določimo barvo:

```
SDL_SetRenderDrawColor(renderer, 255, 0, 0, 255);
```

in narišemo pravokotnik:

```
SDL_RenderFillRect(renderer, &square);
```

Pomembno je, da risanje izvedemo pred:

```
SDL_RenderPresent(renderer);
```

Premikanje kvadrata

Ko uporabnik pritisne puščico, želimo spremeniti položaj kvadrata.

```
if (event.type == SDL_EVENT_KEY_DOWN){
    if (event.key.key == SDLK_LEFT)
        square.x -= 10;

    if (event.key.key == SDLK_RIGHT)
        square.x += 10;

    if (event.key.key == SDLK_UP)
        square.y -= 10;

    if (event.key.key == SDLK_DOWN)
        square.y += 10;
}
```

Celoten program

```
#include <SDL3/SDL.h>
int main() {
    SDL_Init(SDL_INIT_VIDEO);
    SDL_Window* window;
    SDL_Renderer* renderer;
    SDL_CreateWindowAndRenderer("SDL program",800,600,0,&window,&renderer);
    SDL_FRect square{ 350.0f, 250.0f, 100.0f, 100.0f };
    bool running = true;
    while (running) {
        SDL_Event event;
        while (SDL_PollEvent(&event)) {
            if (event.type == SDL_EVENT_QUIT) { running = false; }
            if (event.type == SDL_EVENT_KEY_DOWN) {
                if (event.key.key == SDLK_LEFT) square.x -= 10;
                if (event.key.key == SDLK_RIGHT) square.x += 10;
                if (event.key.key == SDLK_UP) square.y -= 10;
                if (event.key.key == SDLK_DOWN) square.y += 10;
            }
        }
        SDL_SetRenderDrawColor( renderer, 30, 30, 30, 255 );
        SDL_RenderClear(renderer);
        SDL_SetRenderDrawColor( renderer, 255, 0, 0, 255 );
        SDL_RenderFillRect( renderer, &square );
        SDL_RenderPresent(renderer);
    }
    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();
    return 0;
}
```

Naloge

- Spremenite barvo kvadrata.
- Spremenite velikost kvadrata.
- Dodajte omejitev, da kvadrat ne more zapustiti okna.
- Dodajte še en kvadrat.
  - Prvi naj se premika s puščicami.
  - Drugi nas se premika tipkami: W, A, S in D
- Dodajte zaznavo trčenja med kvadratoma. Na začetku sta na različnih položajih.

Premikanje s pomočjo igralnega ploščka.

```
...
SDL_Init(SDL_INIT_VIDEO | SDL_INIT_GAMEPAD);
...
SDL_Gamepad* gamepad = nullptr;
int count = 0;
SDL_JoystickID* joysticks = SDL_GetGamepads(&count);
if (count > 0) gamepad = SDL_OpenGamepad(joysticks[0]);
SDL_free(joysticks);
if (!gamepad) std::cout << "Kontroler ni prikljucen.\n";
...
if (gamepad){
    Sint16 x = SDL_GetGamepadAxis(gamepad, SDL_GAMEPAD_AXIS_LEFTX);
    Sint16 y = SDL_GetGamepadAxis(gamepad, SDL_GAMEPAD_AXIS_LEFTY);

    float dx = x / 32768.0f;
    float dy = y / 32768.0f;
    float speed = 0.1f;

    square.x += dx * speed;
    square.y += dy * speed;

    if (square.x < 0) square.x = 0;
    if (square.y < 0) square.y = 0;
    if (square.x > 700) square.x = 700;
    if (square.y > 500) square.y = 500;
}
...
if (gamepad) SDL_CloseGamepad(gamepad);
```