Umetnost programiranja v C++

Borko Bošković, Blaž Pšeničnik

21. avgust 2025

Uvod

Spoznali bomo

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

Termini delavnice (učilnica G219)

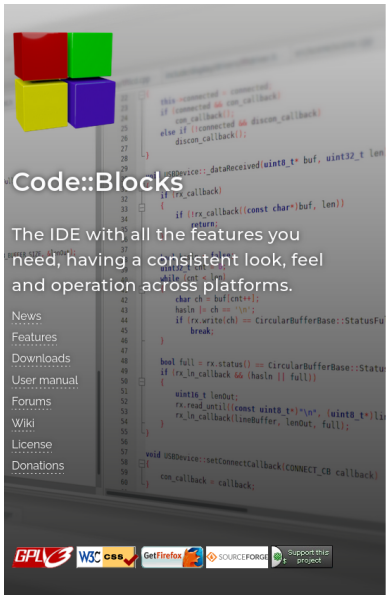
1. 20. 8. 2025 od 9:00 do 15:00
2. 21. 8. 2025 od 9:00 do 15:00
3. 22. 8. 2025 od 9:00 do 15:00

Dokumenti

1. Izpolnjeno in podpisano potrdilo s strani staršev oz. skrbnikov.
2. Seznam prisotnosti
3. Boni za malico in sladoled

Razvojno okolje

Namestitev Code::Blocks



Code::Blocks / Downloads / Binary releases

Binary releases

Please select a setup package depending on your platform:

- Windows XP / Vista / 7 / 8.x / 10
- Linux 32 and 64-bit
- Mac OS X

NOTE: For older OSes use older releases. There are releases for many OS version and platforms on the **Sourceforge.net** page.

NOTE: There are also more recent nightly builds available in the **forums** or (for Ubuntu users) in the **Ubuntu PPA repository**. Please note that we consider nightly builds to be stable, usually.

NOTE: We have a **Changelog for 25.03**, that gives you an overview over the enhancements and fixes we have put in the new release.

NOTE: The default builds are 64 bit (starting with release 20.03). We also provide 32bit builds only for convenience.

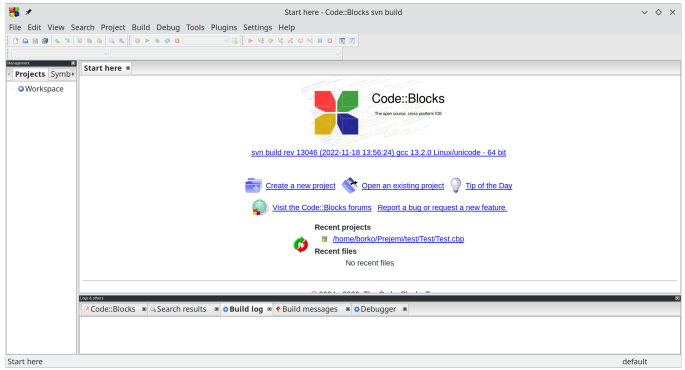
Microsoft Windows (64 bit, default)

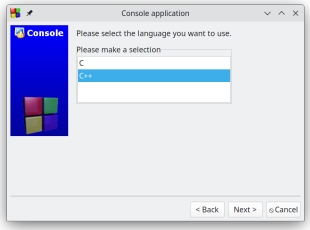
File	Download from
codeblocks-25.03-setup.exe	Sourceforge.net
codeblocks-25.03-setup-nonadmin.exe	Sourceforge.net
codeblocks-25.03-nosetup.zip	Sourceforge.net
codeblocks-25.03mingw-setup.exe	Sourceforge.net
codeblocks-25.03mingw-nosetup.zip	Sourceforge.net

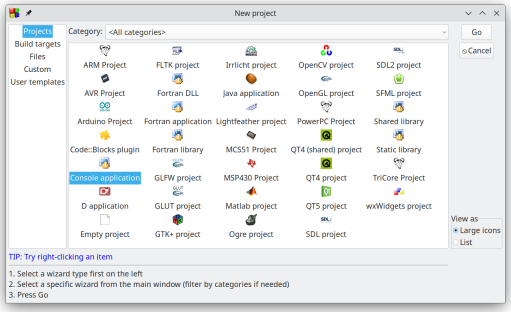
NOTE: The codeblocks-25.03-setup.exe file includes Code::Blocks with all plugins. The codeblocks-25.03-setup-nonadmin.exe file is provided for convenience to users that do not have administrator rights on their

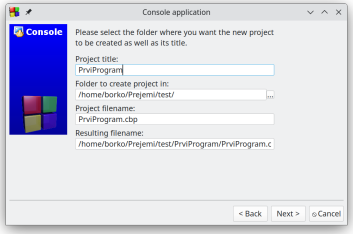
1. `http://www.codeblocks.org/`
2. Download
3. Download the binary release
4. Microsoft Windows (64 bit, default)
5. Prenesi setup.exe
6. Namesti codeblocks in mingw

Ustvarjanje novega projekta



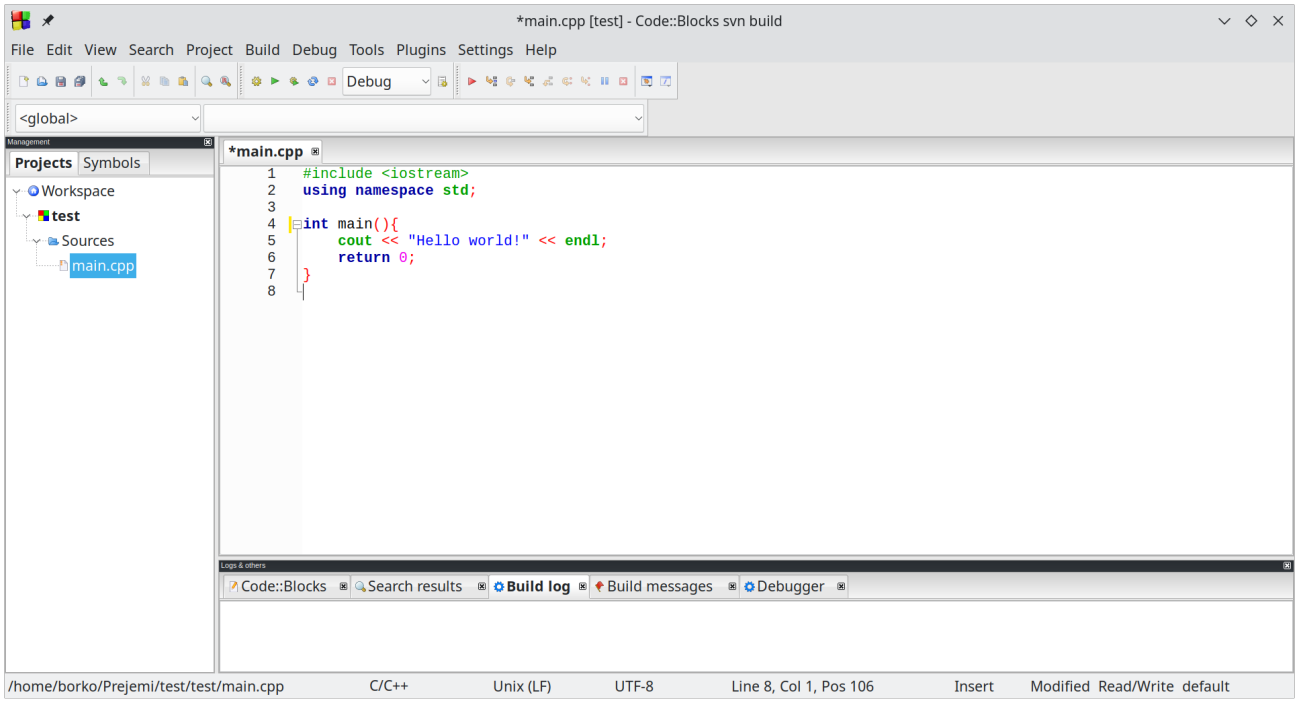






1. Odpri Code::Blocks
2. Klikni **File** → **New** → **Project...**
3. Izberi **Console Application** in klikni **Go**
4. Izberi programski jezik **C++**
5. Nastavi ime in mapo projekta

Prevajanje in zagon



1. Code::Blocks ustvari datoteko `main.cpp` z vzorčno kodo
2. Prevajanje: klikni **Build** (ali pritisni `F9`)
3. Zagon: klikni **Run** (ali pritisni `Ctrl+F10`)
4. Združeno prevajanje in zagon: `F9`
5. Orodjarna

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

```cpp
#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

```cpp
#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.

Grafični vmesnik

Knjižniza za podporo multimediji

- `SFML` (angl. Simple and Fast Multimedia Library)
- Knjižnica za:
  - 2D grafiko
  - obdelavo dogodkov
  - zvok in glasbo
  - omrežno komunikacijo
- Večplatformska (Windows, Linux, macOS)
- Enostavna za začetnike, dovolj zmogljiva za projekte

Namestitev SFML

Linux:

```
sudo apt install libsfml-dev
```

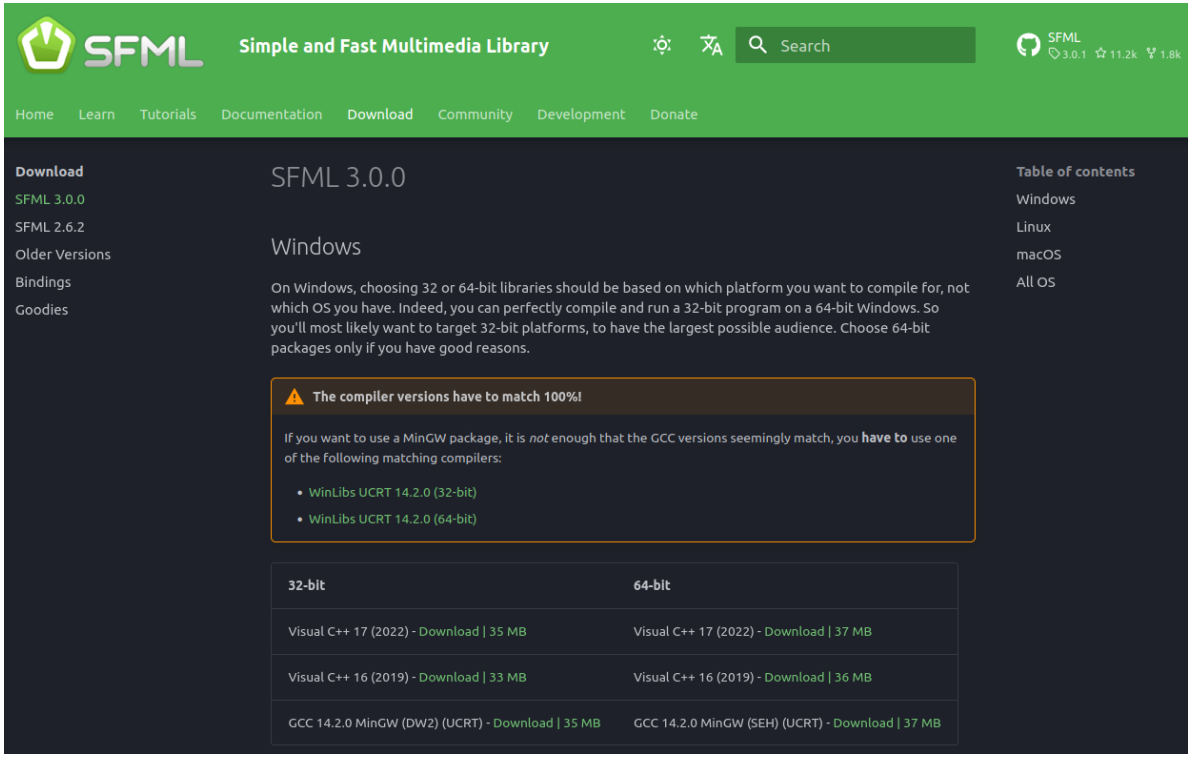
macOS (Homebrew):

```
brew install sfml
```

Windows:

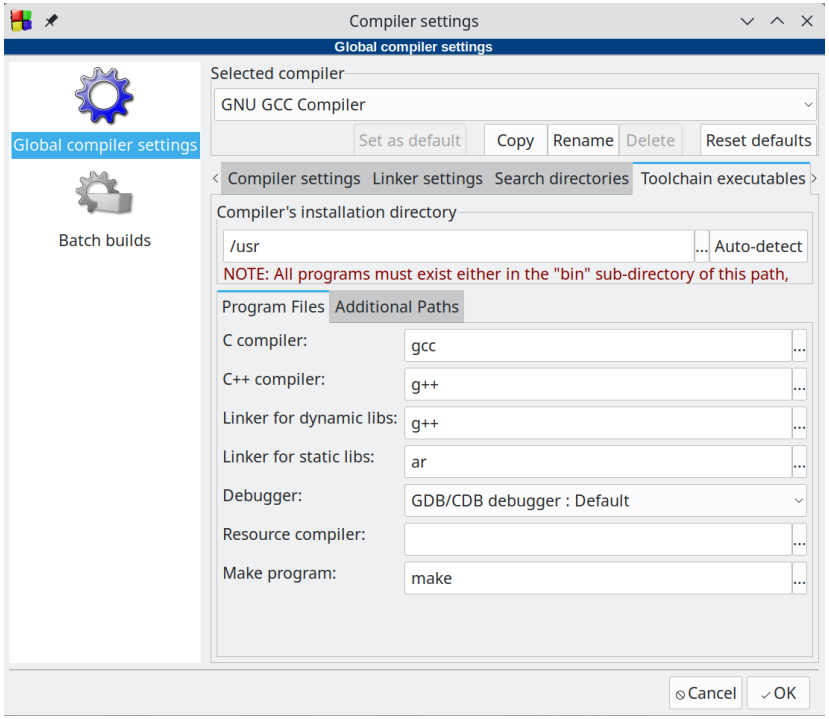
- Prenesi SFML in MinGW-W64 z `https://www.sfml-dev.org`
- Nastavi pot do MinGW-W64 (Code::Blocks)
- Nastavi standard C++17 (Code::Blocks)
- Dodaj `include` in `lib` poti v projekt (Code::Blocks)
- Poveži knjižnice: `sfml-graphics`, `sfml-window`, `sfml-system` (Code::Blocks)
- Kopiraj knjižnic DLL v mapo z izvedljivim programom
- Prenesi fonte z `https://font.download/font/arial`

Prenesi SFML in MinGW-W64



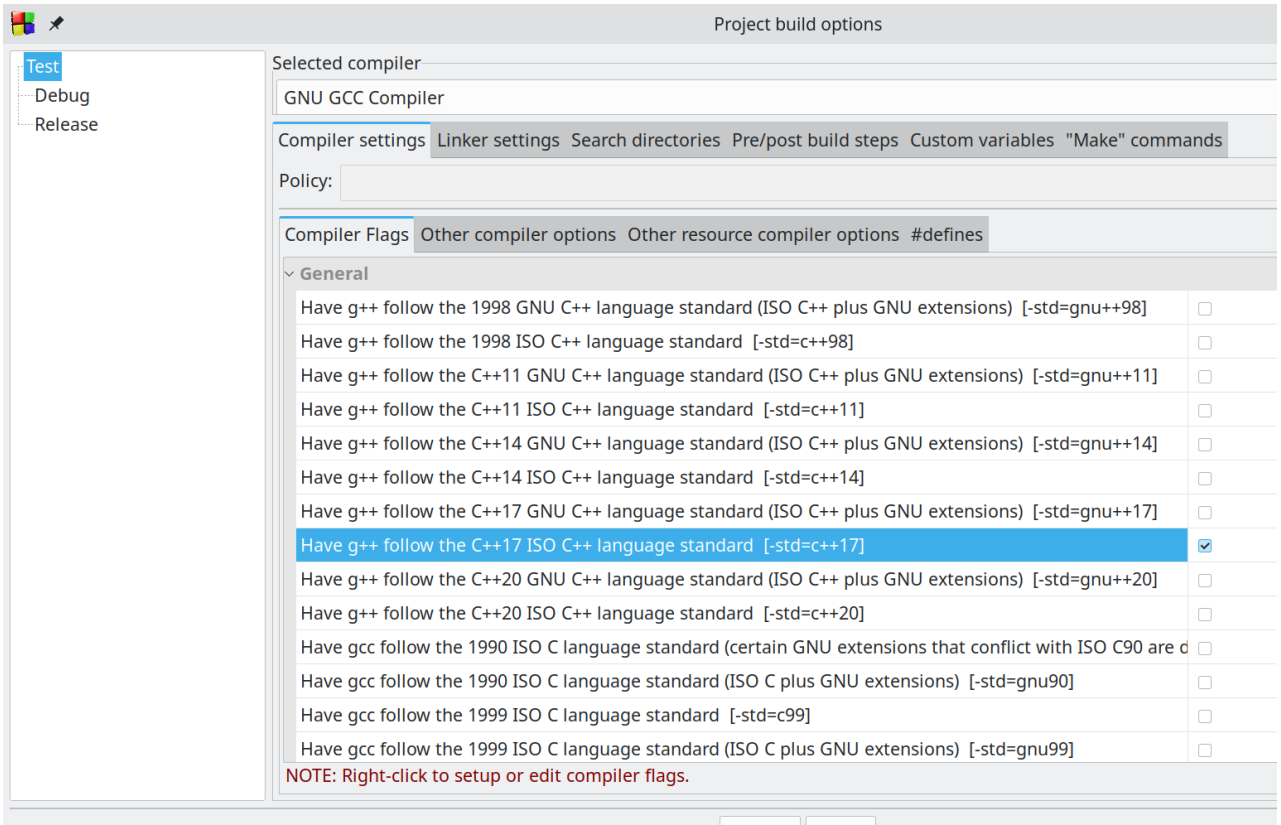
1. `https://www.sfml-dev.org`
2. Download
3. Latest stable version
4. Povezavi:
   - WinLibs UCRT 14.2.0 (64-bit)
   - GCC 14.2.0 MinGW (SEH) (UCRT) (64-bit)
5. Datoteki razširi

Nastavi pot do prevajalnika



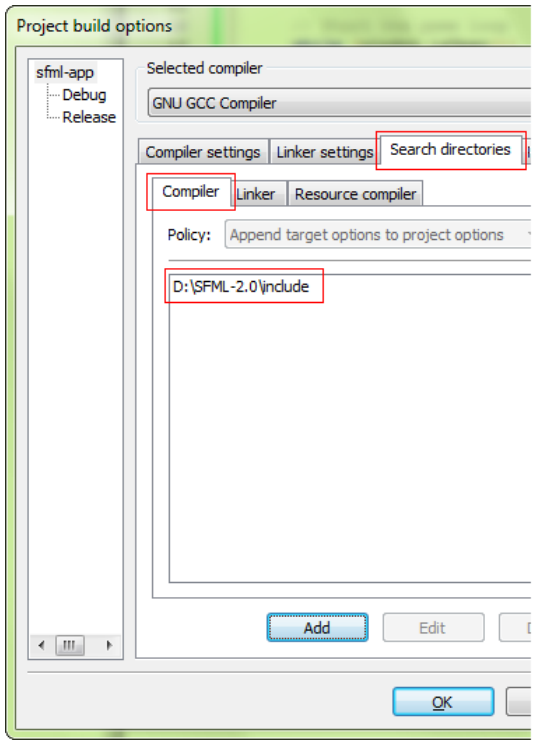
1. Code::Blocks
2. Settings
3. Compiler...
4. Toolchainh executable
5. Compiler's instalation directory: pot do MinGW-W64

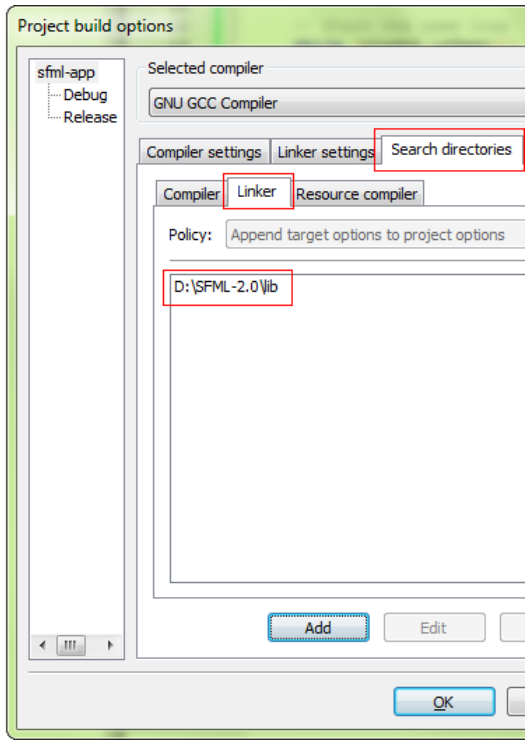
Nastavi standard C++17



1. Code::Blocks
2. Project
3. Build options...
4. Debug/Release
5. Compiler settings
6. Compiler flags
7. -std=c++17

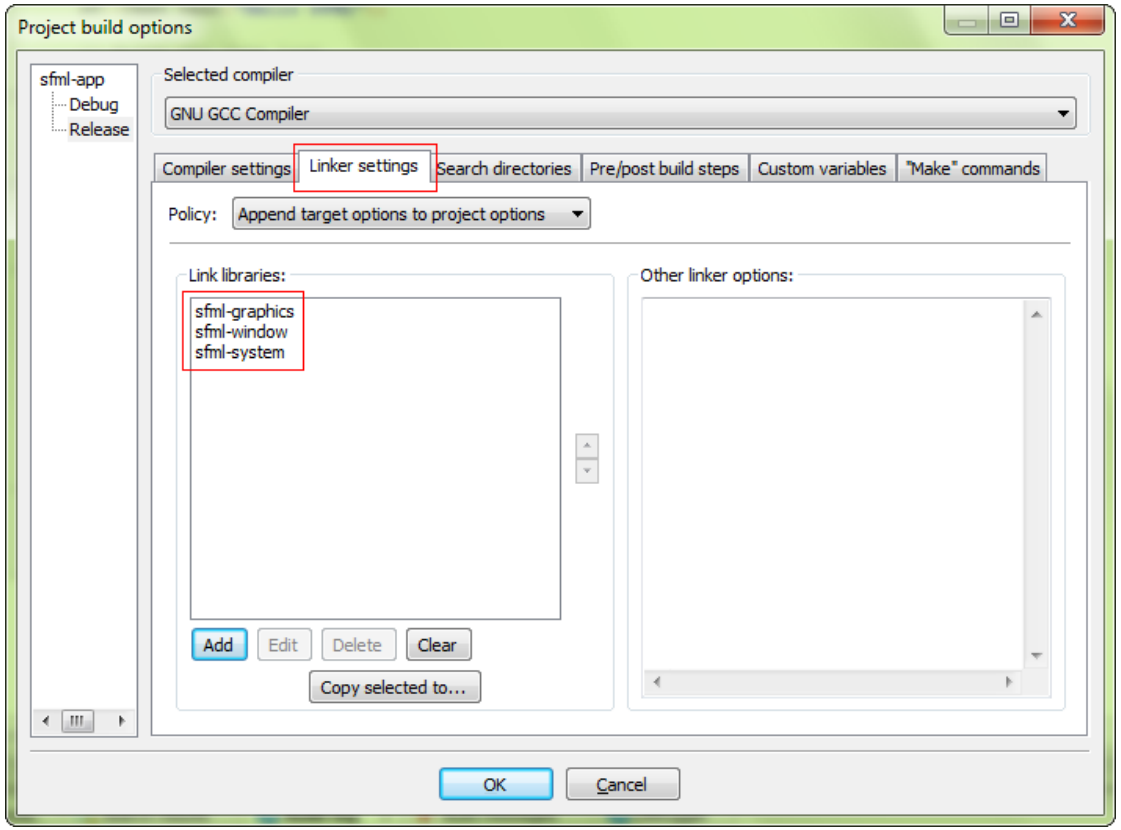
Dodaj `include` in `lib` poti v projekt





1. Code::Blocks
2. Project
3. Build options...
4. Debug/Release
5. Search directories
6. Compiler: Pot do mape **SFLM/include**
7. Linker: Pot do mape **SFLM/lib**

Poveži knjižnice: `sfml-graphics`, `sfml-window`, `sfml-system`



1. Code::Blocks
2. Project
3. Build options...
4. Debug/Release
5. Linker settings
6. Link ribraries:
   - `sfml-graphics`
   - `sfml-window`
   - `sfml-system`

Kopiraj knjižnic DLL

- Po uspešnem prevajanju program (Code::Blocks)
- Kopiramo vsebino `SFML/bin/` v mapo projekta `bin/debug` in `bin/release`
- Program zaženemo (Code::Blocks)

Osnovna struktura SFML programa

```cpp
#include <SFML/Graphics.hpp>

int main() {
    sf::RenderWindow window(sf::VideoMode(800, 600), "SFML Okno");

    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        window.clear(sf::Color::Black);
        // risanje ...
        window.display();
    }
    return 0;
}
```

Osnovni koncepti

- **sf::RenderWindow** — glavno okno za prikaz
- **sf::Event** — obdelava uporabniških dogodkov
- **clear(), draw(), display()** — risanja
- **sf::Shape**, **sf::Sprite** — objekti za prikaz

Igra: Ujemi krog

Cilj igre:

- Premikaj kvadrat (igralca) s tipkami `WASD`
- Ujemi zeleni krog, da dobiš točke
- Cilj se ob ulovu pojavi na naključni lokaciji

Igra: Ujemi krog I

```
#include <SFML/Graphics.hpp>
#include <cstdlib>
#include <ctime>
#include <string>

int main() {
    std::srand(static_cast<unsigned>(std::time(nullptr)));

    sf::RenderWindow window(sf::VideoMode(600, 400), "Ujemi krog - Tocke!");
    window.setFramerateLimit(60);

    // Nalozi pisavo (arial.ttf mora biti v isti mapi)
    sf::Font font;
    if (!font.loadFromFile("arial.ttf")) {
        return -1; // ni nasel pisave
    }

    // Besedilo za tocke
    int score = 0;
    sf::Text scoreText;
    scoreText.setFont(font);
    scoreText.setCharacterSize(24);
    scoreText.setFillColor(sf::Color::White);
    scoreText.setPosition(10, 10);
    scoreText.setString("Tocke: 0");

    // Igralec
    sf::RectangleShape player(sf::Vector2f(40, 40));
```

Igra: Ujemi krog II

```
    player.setFillColor(sf::Color::Red);
    player.setPosition(300, 200);

    // Cilj
    sf::CircleShape target(20);
    target.setFillColor(sf::Color::Green);
    target.setPosition(
        std::rand() % (600 - 40),
        std::rand() % (400 - 40)
    );

    float speed = 4.0f;

    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        // Premikanje igralca
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::W))
            player.move(0, -speed);
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::S))
            player.move(0, speed);
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::A))
            player.move(-speed, 0);
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::D))
```

Igra: Ujemi krog III

```
            player.move(speed, 0);

        // Preveri trk
        if (player.getGlobalBounds().intersects(target.getGlobalBounds())) {
            score++;
            scoreText.setString("Tocke: " + std::to_string(score));
            target.setPosition(
                std::rand() % (600 - 40),
                std::rand() % (400 - 40)
            );
            speed += 0.1f; // pospesimo
        }

        // Risanje
        window.clear(sf::Color::Black);
        window.draw(player);
        window.draw(target);
        window.draw(scoreText);
        window.display();
    }

    return 0;
}
```

Naloge

- Omeji čas igre in izpiši rezultat
- Dodaj več ciljev hkrati
- Dodaj ovire, ki zmanjšajo točke

Naredi svojo igro

Uporabi znanje iz delavnice in ustvari svojo preprosto igro s SFML.

- **Pong** — odbij žogico, da ne pade iz zaslona
- **Snake** — rasteč kačon, ki zbira hrano
- **Space Shooter** — premik vesoljske ladje in streljanje meteoritov
- **Labirint** — premik lika skozi labirint do cilja
- **Memory Game** — ujemanje parov slik

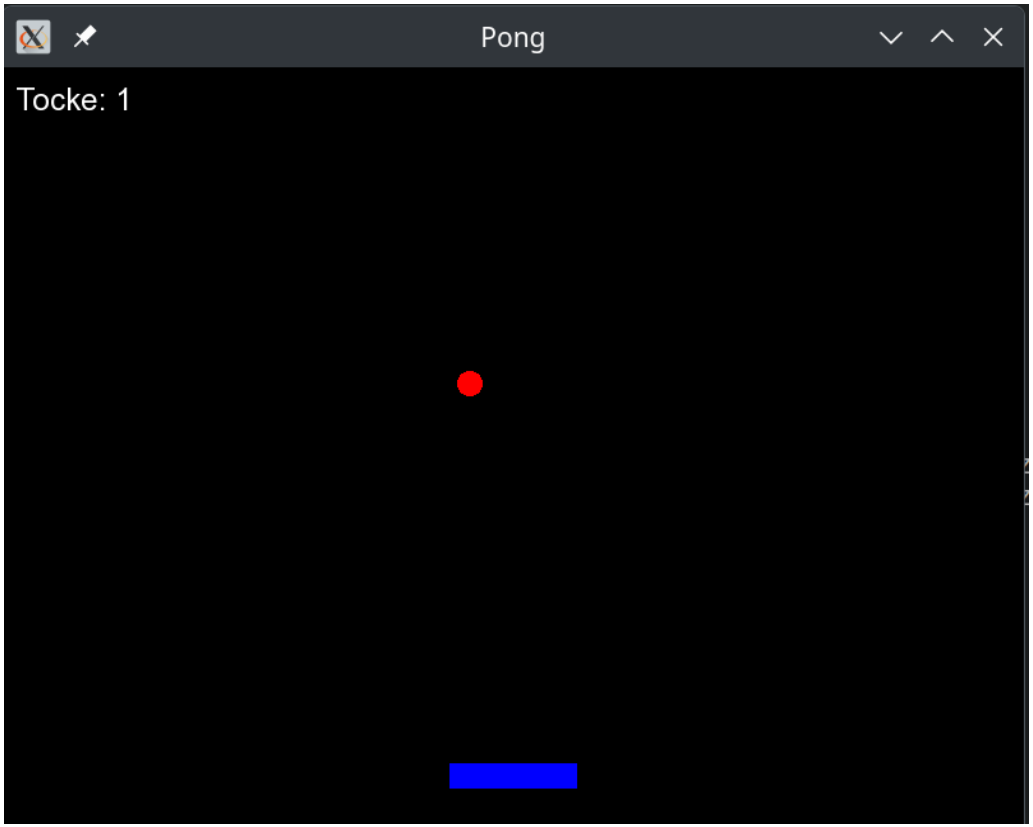
Koraki za izdelavo igre

1. Določi cilj igre
2. Ustvari osnovno okno v SFML
3. Dodaj like (kvadrati, krogi, slike)
4. Dodaj nadzor z tipkovnico ali miško
5. Implementiraj logiko igre (točke, zmaga/poraz)
6. Dodaj grafiko, zvok, animacije
7. Testiraj in izboljšaj

Namigi za uspeh

- Začni z majhnim — najprej delujoča osnova
- Dodaj funkcionalnosti postopoma
- Piši čisto in berljivo kodo
- Uporabi barve, zvoke in animacije za boljšo izkušnjo
- Če obstaneš, poglej dokumentacijo:
  `https://www.sfml-dev.org/documentation`

Igra: Pong



Igra: Pong I

```
#include <SFML/Graphics.hpp>

int main() {
    sf::RenderWindow window(sf::VideoMode(800, 600), "Pong");
    window.setFramerateLimit(60);

    // Igralceva ploscica
    sf::RectangleShape paddle(sf::Vector2f(100, 20));
    paddle.setFillColor(sf::Color::Blue);
    paddle.setPosition(350, 550);

    // Zogica
    sf::CircleShape ball(10);
    ball.setFillColor(sf::Color::Red);
    ball.setPosition(400, 300);

    // Hitrost zogice
    sf::Vector2f ballVelocity(-4.f, -4.f);

    int score = 0;
    sf::Font font;
    if (!font.loadFromFile("arial.ttf")) {
        return -1;
    }
    sf::Text scoreText("Tocke: 0", font, 24);
    scoreText.setFillColor(sf::Color::White);
    scoreText.setPosition(10, 10);
```

Igra: Pong II

```
    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        // Premikanje ploscice
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)
            && paddle.getPosition().x > 0)
            paddle.move(-6.f, 0.f);
        if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right)
            && paddle.getPosition().x < 700)
            paddle.move(6.f, 0.f);

        // Premikanje zogice
        ball.move(ballVelocity);

        // Odboj od leve in desne stene
        if (ball.getPosition().x <= 0 || ball.getPosition().x >= 780)
            ballVelocity.x = -ballVelocity.x;

        // Odboj od zgornje stene
        if (ball.getPosition().y <= 0)
            ballVelocity.y = -ballVelocity.y;

        // Trk s ploscico
        if (ball.getGlobalBounds().intersects(paddle.getGlobalBounds())) {
```

Igra: Pong III

```
            ballVelocity.y = -ballVelocity.y;
            score++;
            scoreText.setString("Tocke: " + std::to_string(score));
        }

        // Ce zogica pade pod ploscico
        if (ball.getPosition().y > 600) {
            score = 0;
            scoreText.setString("Tocke: 0");
            ball.setPosition(400, 300);
            ballVelocity = sf::Vector2f(-4.f, -4.f);
        }

        // Risanje
        window.clear(sf::Color::Black);
        window.draw(paddle);
        window.draw(ball);
        window.draw(scoreText);
        window.display();
    }

    return 0;
}
```

Igra: Snake

Igra: Snake I

```
#include <SFML/Graphics.hpp>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <string>

const int windowWidth = 400;
const int windowHeight = 400;
const int blockSize = 20;

enum Direction { Up, Down, Left, Right };

struct SnakeSegment { int x, y; };

int main() {
    sf::RenderWindow window(sf::VideoMode(windowWidth, windowHeight), "Snake");

    // Nastavitve za kaco
    std::vector<SnakeSegment> snake = { {10,10}, {9,10}, {8,10} };
    Direction dir = Right;
    int score = 0;
    float speed = 0.2f; // casovni interval med premiki (sekunde)
    sf::Clock moveClock;

    // Hrana
    std::srand(std::time(nullptr));
    SnakeSegment food = { std::rand() %
        (windowWidth / blockSize), std::rand() % (windowHeight / blockSize) };
```

Igra: Snake II

```
// Font in besedilo
sf::Font font;
if(!font.loadFromFile("arial.ttf"))  return -1; // font mora biti v isti mapi

sf::Text scoreText;
scoreText.setFont(font);
scoreText.setCharacterSize(20);
scoreText.setFillColor(sf::Color::White);
scoreText.setPosition(10,10);

bool gameOver = false;

while(window.isOpen()) {
    sf::Event event;
    while(window.pollEvent(event)) {
        if(event.type == sf::Event::Closed)
            window.close();
        if(event.type == sf::Event::KeyPressed) {
            if(event.key.code == sf::Keyboard::Up && dir != Down)
                dir = Up;
            if(event.key.code == sf::Keyboard::Down && dir != Up)
                dir = Down;
            if(event.key.code == sf::Keyboard::Left && dir != Right)
                dir = Left;
            if(event.key.code == sf::Keyboard::Right && dir != Left)
                dir = Right;
            if(gameOver && event.key.code == sf::Keyboard::R) {
```

Igra: Snake III

```
                // Reset igre
                snake = { {10,10}, {9,10}, {8,10} };
                dir = Right;
                score = 0;
                speed = 0.2f;
                food = { std::rand() %
                    (windowWidth / blockSize),
                    std::rand() % (windowHeight / blockSize) };
                gameOver = false;
                moveClock.restart();
            }
        }
    }
    if(!gameOver && moveClock.getElapsedTime().asSeconds() >= speed) {
        moveClock.restart();

        // Premikanje kace
        for(int i = snake.size()-1; i>0; --i) {
            snake[i] = snake[i-1];
        }
        switch(dir) {
            case Up: snake[0].y -= 1; break;
            case Down: snake[0].y += 1; break;
            case Left: snake[0].x -= 1; break;
            case Right: snake[0].x += 1; break;
        }
        // Preverjanje kolizije s stenami
        if(snake[0].x < 0 || snake[0].x >= windowWidth/blockSize ||
```

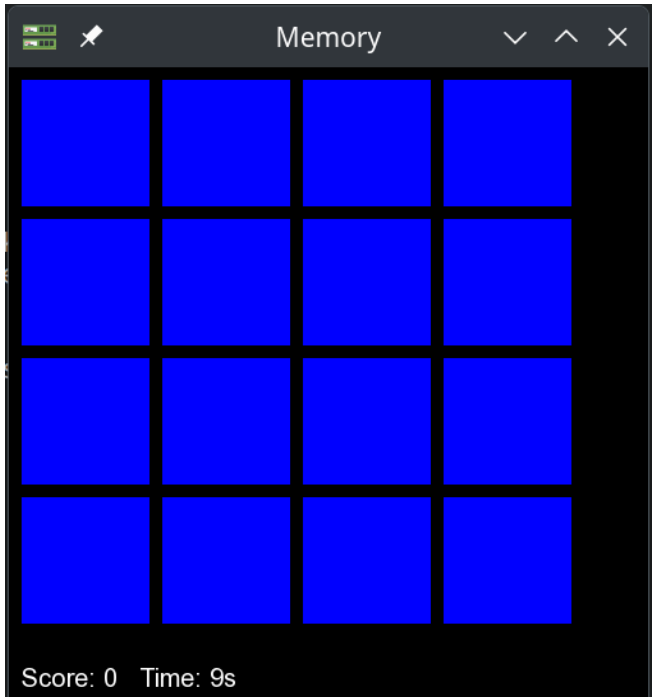
Igra: Snake IV

```
            snake[0].y < 0 || snake[0].y >= windowHeight/blockSize) {
             gameOver = true;
        }
        // Preverjanje kolizije s samim seboj
        for(int i=1;i<snake.size();++i){
            if(snake[0].x == snake[i].x && snake[0].y == snake[i].y)
                gameOver = true;
        }
        // Preverjanje hrane
        if(snake[0].x == food.x && snake[0].y == food.y) {
            snake.push_back({-1,-1}); // dodaj segment
            food.x = std::rand() % (windowWidth / blockSize);
            food.y = std::rand() % (windowHeight / blockSize);
            score += 10;
            if(speed > 0.05f) speed -= 0.01f; // povecaj hitrost
        }
    }
    // Risanje
    window.clear(sf::Color::Black);

    sf::RectangleShape block(sf::Vector2f(blockSize, blockSize));
    block.setFillColor(sf::Color::Green);
    for(auto &seg: snake){
        block.setPosition(seg.x*blockSize, seg.y*blockSize);
        window.draw(block);
    }
    block.setFillColor(sf::Color::Red);
    block.setPosition(food.x*blockSize, food.y*blockSize);
```

Igra: Snake V

```
            window.draw(block);

            scoreText.setString("Score: " + std::to_string(score));
            window.draw(scoreText);

            if(gameOver){
                sf::Text goText;
                goText.setFont(font);
                goText.setCharacterSize(30);
                goText.setFillColor(sf::Color::Yellow);
                goText.setString("GAME OVER\nPress R to restart");
                goText.setPosition(50, windowHeight/2 - 50);
                window.draw(goText);
            }

            window.display();
        }
        return 0;
}
```

Igra: Memory



Igra: Memory I

```cpp
#include <SFML/Graphics.hpp>
#include <vector>
#include <ctime>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <random>

const int windowWidth = 500;
const int windowHeight = 500;
const int gridSize = 4; // 4x4 mreza
const int cardSize = 100;
const int padding = 10;

struct Card {
    int value;
    bool revealed;
    bool matched;
    sf::RectangleShape shape;
    sf::Text text;
};

int main() {
    std::mt19937 rand;
    sf::RenderWindow window(sf::VideoMode(windowWidth, windowHeight), "Memory");

    sf::Font font;
    if(!font.loadFromFile("arial.ttf")) return -1;
```

Igra: Memory II

```
// Ustvarjanje kart
std::vector<int> values;
for(int i=0; i<(gridSize*gridSize)/2; ++i){
    values.push_back(i);
    values.push_back(i);
}
std::srand(std::time(nullptr));
std::shuffle(values.begin(), values.end(), rand);

std::vector<Card> cards;
int idx = 0;
for(int y=0; y<gridSize; ++y){
    for(int x=0; x<gridSize; ++x){
        Card c;
        c.value = values[idx++];
        c.revealed = false;
        c.matched = false;
        c.shape = sf::RectangleShape(sf::Vector2f(cardSize, cardSize));
        c.shape.setFillColor(sf::Color::Blue);
        c.shape.setPosition(x*(cardSize+padding)+padding,
                            y*(cardSize+padding)+padding);

        c.text.setFont(font);
        c.text.setString(std::to_string(c.value));
        c.text.setCharacterSize(40);
        c.text.setFillColor(sf::Color::White);
        c.text.setPosition(c.shape.getPosition().x+cardSize/4,
```

Igra: Memory III

```
                                c.shape.getPosition().y+cardSize/4);

            cards.push_back(c);
        }
    }
    Card* first = nullptr;
    Card* second = nullptr;
    sf::Clock revealClock;
    int score = 0;
    sf::Clock gameClock;

    sf::Text scoreText;
    scoreText.setFont(font);
    scoreText.setCharacterSize(20);
    scoreText.setFillColor(sf::Color::White);
    scoreText.setPosition(10, windowHeight - 30);

    while(window.isOpen()){
        sf::Event event;
        while(window.pollEvent(event)){
            if(event.type == sf::Event::Closed) window.close();

            if(event.type == sf::Event::MouseButtonPressed &&
                event.mouseButton.button == sf::Mouse::Left){
                sf::Vector2f mousePos(event.mouseButton.x, event.mouseButton.y);
                for(auto &c: cards){
                    if(c.shape.getGlobalBounds().contains(mousePos) &&
                        !c.revealed && !c.matched){
```

Igra: Memory IV

```
                    if(!first) first = &c;
                    else if(!second && &c != first) second = &c;
                    c.revealed = true;
                    revealClock.restart();
                    break;
                }
            }
        }
    }
    // Preverjanje ujemanja
    if(first && second && revealClock.getElapsedTime().asSeconds() > 1.0f){
        if(first->value == second->value){
            first->matched = true;
            second->matched = true;
            score += 10;
        } else {
            first->revealed = false;
            second->revealed = false;
        }
        first = nullptr;
        second = nullptr;
    }
    // Risanje
    window.clear(sf::Color::Black);

    for(auto &c: cards){
        window.draw(c.shape);
        if(c.revealed || c.matched) window.draw(c.text);
```

Igra: Memory V

```
        }
        scoreText.setString("Score: " + std::to_string(score) + "   Time: " +
            std::to_string((int)gameClock.getElapsedTime().asSeconds()) + "s");
        window.draw(scoreText);

        window.display();

        // Preveri konec igre
        bool allMatched = true;
        for(auto &c: cards){
            if(!c.matched){
                allMatched = false;
                break;
            }
        }
        if(allMatched){
            sf::Text endText;
            endText.setFont(font);
            endText.setCharacterSize(40);
            endText.setFillColor(sf::Color::Yellow);
            endText.setString("You Win!\nScore: " +
                std::to_string(score) + "\nPress R to restart");
            endText.setPosition(50, windowHeight/2 - 50);
            window.draw(endText);
            window.display();

            bool restart = false;
            while(!restart){
```

Igra: Memory VI

```
                while(window.pollEvent(event)){
                    if(event.type == sf::Event::Closed) window.close();
                    if(event.type == sf::Event::KeyPressed &&
                        event.key.code == sf::Keyboard::R){
                        restart = true;
                    }
                }
            }
            // Reset igre
            std::shuffle(values.begin(), values.end(), rand);
            idx = 0;
            for(int i=0;i<cards.size();++i){
                cards[i].value = values[idx++];
                cards[i].revealed = false;
                cards[i].matched = false;
                cards[i].text.setString(std::to_string(cards[i].value));
            }
            score = 0;
            gameClock.restart();
        }
    }

    return 0;
}
```

Igra: Maze

Igra: Labirint I

```cpp
#include <SFML/Graphics.hpp>
#include <vector>
#include <sstream>

int main() {
    const int cellSize = 40;
    const int rows = 10;
    const int cols = 10;

    // 0 = pot, 1 = zid, 2 = tocka
    int maze[rows][cols] = {
        {0,1,0,0,2,1,0,0,0,0},
        {0,1,0,1,0,1,2,1,1,0},
        {0,0,0,1,0,0,0,1,0,0},
        {1,1,0,1,1,1,0,1,0,1},
        {0,0,2,0,0,0,0,1,0,0},
        {0,1,1,1,1,1,0,1,1,0},
        {0,0,0,0,0,0,0,0,1,0},
        {0,1,0,1,1,1,1,0,1,0},
        {0,1,0,0,0,0,0,0,1,2},
        {0,0,0,1,1,1,1,0,0,0}
    };

    sf::Vector2i goal(9, 9);
    sf::Vector2i playerPos(0, 0);
    int score = 0;
    bool gameWon = false;
```

Igra: Labirint II

```
    sf::RenderWindow window(sf::VideoMode(cols * cellSize, rows * cellSize),
                            "Maze");
    window.setFramerateLimit(10);

    sf::Font font;
    font.loadFromFile("arial.ttf");
    sf::Text info;
    info.setFont(font);
    info.setCharacterSize(20);
    info.setFillColor(sf::Color::Yellow);

    while (window.isOpen()) {
        sf::Event e;
        while (window.pollEvent(e))
            if (e.type == sf::Event::Closed) window.close();

        if (!gameWon) {
            sf::Vector2i move(0, 0);
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Up)) move = {0, -1};
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Down)) move = {0, 1};
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left)) move = {-1, 0};
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right)) move = {1, 0};

            sf::Vector2i newPos = playerPos + move;
            if (newPos.x >= 0 && newPos.x < cols && newPos.y >= 0 &&
                newPos.y < rows && maze[newPos.y][newPos.x] != 1) {
                playerPos = newPos;
```

Igra: Labirint III

```
            if (maze[playerPos.y][playerPos.x] == 2) {
                score++;
                maze[playerPos.y][playerPos.x] = 0;
            }
            if (playerPos == goal) {
                gameWon = true;
            }
        }
    }

    window.clear();
    sf::RectangleShape cell(sf::Vector2f(cellSize - 1, cellSize - 1));

    for (int y = 0; y < rows; y++) {
        for (int x = 0; x < cols; x++) {
            if (maze[y][x] == 1)
                cell.setFillColor(sf::Color::Black);
            else if (maze[y][x] == 2)
                cell.setFillColor(sf::Color::Green);
            else
                cell.setFillColor(sf::Color::White);

            cell.setPosition(x * cellSize, y * cellSize);
            window.draw(cell);
        }
    }

    cell.setFillColor(sf::Color::Blue);
```

Igra: Labirint IV

```
            cell.setPosition(playerPos.x * cellSize, playerPos.y * cellSize);
            window.draw(cell);

            cell.setFillColor(sf::Color::Red);
            cell.setPosition(goal.x * cellSize, goal.y * cellSize);
            window.draw(cell);

            std::ostringstream ss;
            if (gameWon)
                ss << "Zmaga! Tocke: " << score;
            else
                ss << "Tocke: " << score;
            info.setString(ss.str());
            info.setPosition(5, 5);
            window.draw(info);

            window.display();
        }
    }
```

Igra: Shooter

Igra: Shooter I

```
#include <SFML/Graphics.hpp>
#include <vector>
#include <sstream>

int main() {
    sf::RenderWindow window(sf::VideoMode(800, 600), "Space Shooter");
    window.setFramerateLimit(60);

    sf::RectangleShape player(sf::Vector2f(50, 20));
    player.setFillColor(sf::Color::Cyan);
    player.setPosition(375, 550);

    sf::RectangleShape bullet(sf::Vector2f(5, 15));
    bullet.setFillColor(sf::Color::Yellow);
    bool bulletActive = false;

    std::vector<sf::RectangleShape> enemies;
    for (int i = 0; i < 5; i++) {
        sf::RectangleShape enemy(sf::Vector2f(40, 20));
        enemy.setFillColor(sf::Color::Red);
        enemy.setPosition(rand() % 760, rand() % 200);
        enemies.push_back(enemy);
    }

    sf::Font font;
    font.loadFromFile("arial.ttf");
    sf::Text scoreText;
    scoreText.setFont(font);
```

Igra: Shooter II

```
    scoreText.setCharacterSize(20);
    scoreText.setFillColor(sf::Color::White);

    int score = 0;
    bool gameOver = false;

    while (window.isOpen()) {
        sf::Event e;
        while (window.pollEvent(e))
            if (e.type == sf::Event::Closed) window.close();

        if (!gameOver) {
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Left) &&
                player.getPosition().x > 0)
                player.move(-5.f, 0.f);
            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Right) &&
                player.getPosition().x < 750)
                player.move(5.f, 0.f);

            if (sf::Keyboard::isKeyPressed(sf::Keyboard::Space) &&
                !bulletActive) {
                bullet.setPosition(player.getPosition().x + 22,
                                   player.getPosition().y - 15);
                bulletActive = true;
            }

            if (bulletActive) {
                bullet.move(0.f, -7.f);
```

Igra: Shooter III

```
            if (bullet.getPosition().y < 0)
                bulletActive = false;
        }

        // Premik sovraznikov
        for (auto& enemy : enemies) {
            enemy.move(0.f, 2.f);
            if (enemy.getPosition().y > 600) {
                enemy.setPosition(rand() % 760, -20);
                gameOver = true; // sovraznik je presel igralca
            }
        }

        // Zadetek
        for (auto& enemy : enemies) {
            if (bulletActive &&
                bullet.getGlobalBounds().intersects(enemy.getGlobalBounds())) {
                enemy.setPosition(rand() % 760, -20);
                bulletActive = false;
                score++;
                // Dodaj novega sovraznika vsakih 5 tock
                if (score % 5 == 0) {
                    sf::RectangleShape newEnemy(sf::Vector2f(40, 20));
                    newEnemy.setFillColor(sf::Color::Red);
                    newEnemy.setPosition(rand() % 760, rand() % 200);
                    enemies.push_back(newEnemy);
                }
                break;
```

Igra: Shooter IV

```
                }
            }
        }

        std::ostringstream ss;
        if (!gameOver)
            ss << "Score: " << score;
        else
            ss << "GAME OVER - Final score: " << score;
        scoreText.setString(ss.str());

        window.clear();
        window.draw(player);
        if (bulletActive) window.draw(bullet);
        for (auto& enemy : enemies)
            window.draw(enemy);
        window.draw(scoreText);
        window.display();
    }
}
```

Obdelava slik

Obdelava slik I

```
#include <SFML/Graphics.hpp>
#include <iostream>
#include <cstdlib>
#include <ctime>

sf::Image toGrayscale(const sf::Image& src) {
    sf::Image img = src;
    for (unsigned y = 0; y < img.getSize().y; ++y) {
        for (unsigned x = 0; x < img.getSize().x; ++x) {
            sf::Color c = img.getPixel(x, y);
            unsigned char g = 0.299*c.r + 0.587*c.g + 0.114*c.b;
            img.setPixel(x, y, sf::Color(g, g, g, c.a));
        }
    }
    return img;
}

sf::Image toNegative(const sf::Image& src) {
    sf::Image img = src;
    for (unsigned y = 0; y < img.getSize().y; ++y) {
        for (unsigned x = 0; x < img.getSize().x; ++x) {
            sf::Color c = img.getPixel(x, y);
            img.setPixel(x, y, sf::Color(255-c.r, 255-c.g, 255-c.b, c.a));
        }
    }
    return img;
}
```

Obdelava slik II

```
sf::Image toSepia(const sf::Image& src) {
    sf::Image img = src;
    for (unsigned y = 0; y < img.getSize().y; ++y) {
        for (unsigned x = 0; x < img.getSize().x; ++x) {
            sf::Color c = img.getPixel(x, y);
            unsigned char tr = std::min(255.0, 0.393*c.r + 0.769*c.g + 0.189*c.b);
            unsigned char tg = std::min(255.0, 0.349*c.r + 0.686*c.g + 0.168*c.b);
            unsigned char tb = std::min(255.0, 0.272*c.r + 0.534*c.g + 0.131*c.b);
            img.setPixel(x, y, sf::Color(tr, tg, tb, c.a));
        }
    }
    return img;
}

sf::Image mirrorHorizontal(const sf::Image& src) {
    sf::Image img = src;
    unsigned w = img.getSize().x;
    unsigned h = img.getSize().y;
    for (unsigned y = 0; y < h; ++y) {
        for (unsigned x = 0; x < w/2; ++x) {
            sf::Color left = img.getPixel(x, y);
            sf::Color right = img.getPixel(w-1-x, y);
            img.setPixel(x, y, right);
            img.setPixel(w-1-x, y, left);
        }
    }
    return img;
}
```

Obdelava slik III

```
sf::Image glitchEffect(const sf::Image& src) {
    sf::Image img = src;
    unsigned w = img.getSize().x;
    unsigned h = img.getSize().y;
    for (int i = 0; i < 1000; i++) {
        unsigned x1 = rand() % w;
        unsigned y1 = rand() % h;
        unsigned x2 = rand() % w;
        unsigned y2 = rand() % h;
        img.setPixel(x1, y1, src.getPixel(x2, y2));
    }
    return img;
}

int main() {
    srand(static_cast<unsigned>(time(nullptr)));

    sf::Image original;
    if (!original.loadFromFile("slika.png")) {
        std::cout << "Napaka pri nalaganju slike!" << std::endl;
        return -1;
    }

    sf::Texture texture;
    texture.loadFromImage(original);
    sf::Sprite sprite(texture);
```

Obdelava slik IV

```
    sf::RenderWindow window(sf::VideoMode(original.getSize().x,
                                          original.getSize().y),
                            "Mini Foto-laboratorij");
    sf::Image current = original;

    while (window.isOpen()) {
        sf::Event e;
        while (window.pollEvent(e)) {
            if (e.type == sf::Event::Closed)
                window.close();

            if (e.type == sf::Event::KeyPressed) {
                if (e.key.code == sf::Keyboard::Num1) current = original;
                else if (e.key.code == sf::Keyboard::Num2)
                    current = toGrayscale(original);
                else if (e.key.code == sf::Keyboard::Num3)
                    current = toNegative(original);
                else if (e.key.code == sf::Keyboard::Num4)
                    current = toSepia(original);
                else if (e.key.code == sf::Keyboard::Num5)
                    current = mirrorHorizontal(original);
                else if (e.key.code == sf::Keyboard::Num6)
                    current = glitchEffect(original);

                texture.loadFromImage(current);
                sprite.setTexture(texture);
            }
        }
```

Obdelava slik V

```
            window.clear();
            window.draw(sprite);
            window.display();
        }

        return 0;
}
```

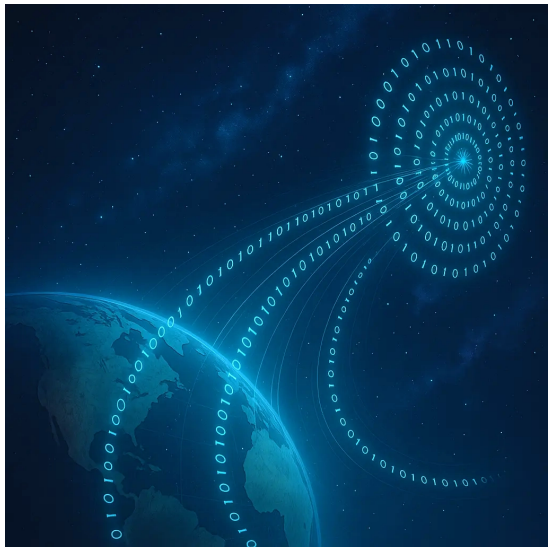
Zaporedja

Binarna zaporedja z nizkimi avtokorelacijami

- LABS = **Low Autocorrelation Binary Sequences**
- Binarna zaporedja dolžine N: vsak element $s_i \in \{+1, -1\}$
- Cilj: zaporedja z majhnimi vrednostmi **izvenničelne avtokorelacije**
- Pomembna v komunikacijah, radarskih sistemih, kriptografiji

Uporaba LABS

- Radar in sonar — zmanjšanje šuma in interference
- Brezžične komunikacije — boljša ločljivost signalov
- Kriptografija — lastnosti podobne naključnim zaporedjem
- Merilni sistemi — večja natančnost pri korelacijskih metodah



Avtokorelacija

Definicija avtokorelacije

Za zamik k:

$$C_k = \sum_{i=1}^{N-k} s_i \cdot s_{i+k}$$

- $C_0 = N$ (največja korelacija pri zamiku 0)
- Pri LABS nas zanimajo C_k za $k \geq 1$
- Nizka avtokorelacija $\Rightarrow$ boljše lastnosti signala

Energetska funkcija

Definicija energije

$$E(S) = \sum_{k=1}^{N-1} C_k^2$$

- Nižja energija $\Rightarrow$ manjša avtokorelacija
- Problem LABS: najti S z minimalno energijo
- Kombinatorično težek problem (eksponentno število možnosti)

Primer

Naj bo $S = (+1, +1, -1, +1)$, $N = 4$:

$$C_1 = (1)(1) + (1)(-1) + (-1)(1) = 1 - 1 - 1 = -1$$

$$C_2 = (1)(-1) + (1)(1) = -1 + 1 = 0$$

$$C_3 = (1)(1) = 1$$

$$E = (-1)^2 + 0^2 + 1^2 = 1 + 0 + 1 = 2$$

Šum v podatkih I

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <cstdlib> // rand, srand
#include <ctime>   // time

// Enostavna LABS sekvenca (Barker-7)
const std::vector<int> LABS = {+1, +1, +1, -1, -1, +1, -1};

// Funkcija za kodiranje bita (0 ali 1)
std::vector<int> encodeBit(int bit) {
    int m = (bit == 0) ? -1 : +1;
    std::vector<int> encoded;
    for(int s : LABS) encoded.push_back(m * s);
    return encoded;
}

// Funkcija za dekodiranje bitov iz LABS zaporedja
int decodeBit(const std::vector<int>& received) {
    int corr = 0;
    for(size_t i=0;i<LABS.size();++i) corr += received[i]*LABS[i];
    return (corr >= 0) ? 1 : 0;
}

// Pretvorba znaka v bitni vektor
std::vector<int> charToBits(char c) {
    std::vector<int> bits(8);
    for(int i=0;i<8;++i) bits[7-i] = (c >> i) & 1;
```

Šum v podatkih II

```
    return bits;
}

// Pretvorba bitov nazaj v znak
char bitsToChar(const std::vector<int>& bits) {
    char c = 0;
    for(int i=0;i<8;++i) c |= (bits[i] << (7-i));
    return c;
}

int main() {
    srand(time(0));

    std::string text = "Franca";
    std::cout << "Original text: " << text << "\n";

    // Zakodiraj besedilo
    std::vector<int> encodedSequence;
    for(char c : text) {
        std::vector<int> bits = charToBits(c);
        for(int b : bits) {
            std::vector<int> enc = encodeBit(b);
            encodedSequence.insert(encodedSequence.end(), enc.begin(), enc.end());
        }
    }

    // Dodamo sum: nakljucno preflipamo 10% elementov
    std::vector<int> noisySequence = encodedSequence;
```

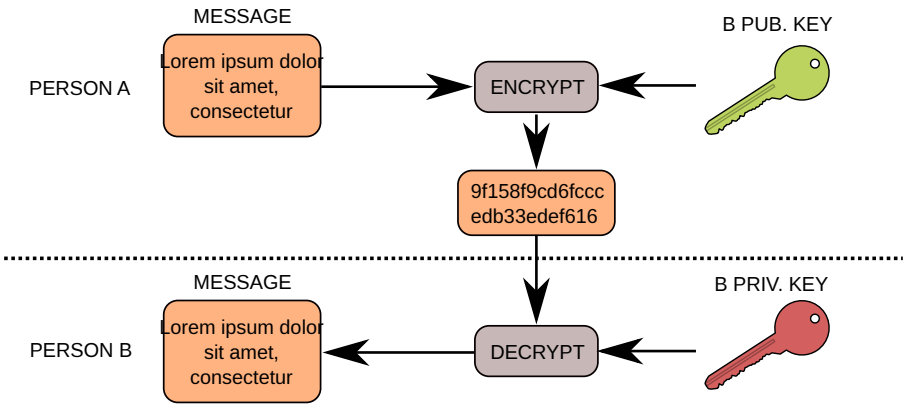
Šum v podatkih III

```
    for(size_t i=0;i<noisySequence.size();++i){
        if(rand()%10 == 0) noisySequence[i] *= -1;
    }

    // Dekodiranje
    std::vector<int> decodedBits;
    for(size_t i=0;i<noisySequence.size();i+=LABS.size()){
        std::vector<int> chunk(noisySequence.begin()+i,
                               noisySequence.begin()+i+LABS.size());
        decodedBits.push_back(decodeBit(chunk));
    }

    // Pretvorba bitov v znake
    std::string decodedText;
    for(size_t i=0;i<decodedBits.size();i+=8){
        std::vector<int> charBits(decodedBits.begin()+i, decodedBits.begin()+i+8);
        decodedText += bitsToChar(charBits);
    }

    std::cout << "Decoded text (with noise): " << decodedText << "\n";

    return 0;
}
```

Algoritem RSA

Kaj je RSA?

- Je star a še vedno uporabljen algoritem
- Razvit leta 1977
- Asimetrični kriptografski algoritem
- Uporablja par ključev: **javni** (n, e) in **zasebni** (n, d)
- Varnost temelji na težavnosti faktorizacije velikega števila n
- Izumili Rivest, Shamir in Adleman leta 1977



Šifriranje in dešifriranje

Šifriranje:

$$c = m^e \bmod n$$

Dešifriranje:

$$m = c^d \bmod n$$

kjer:

- m je izvorno sporočilo (kot številka)
- c je šifrirano sporočilo
- e, n javni ključ
- d, n zasebni ključ

Primer z majhnimi števili

- $p = 61, \quad q = 53$
- $n = 61 \cdot 53 = 3233$
- $\varphi(n) = 60 \cdot 52 = 3120$
- Izberi $e = 17$
- Izračunaj d: $17 \cdot d \equiv 1 \pmod{3120}$, $d = 2753$

Šifriranje: $m = 65$

$$c = 65^{17} \bmod 3233 = 2790$$

Dešifriranje:

$$m = 2790^{2753} \bmod 3233 = 65$$

Kaj je `modPow`?

- `modPow(base, exp, n)` izračuna:

$$r = (\text{base}^{\text{exp}}) \bmod \text{n}$$

- Uporablja se v algoritmu RSA za:
  - Šifriranje: $c = m^e \bmod n$
  - Dešifriranje: $m = c^d \bmod n$
- Ne izračunamo najprej base^{exp} (preveliko število), ampak uporabimo **hitro potenciranje z modulom**.

Ključna lastnost:

$$(a \cdot b) \bmod n = \left[(a \bmod n) \cdot (b \bmod n)\right] \bmod n$$

Primeri rekurzije:

- Če je e **sodo**:

$$a^e \bmod n = \left(a^{e/2} \bmod n\right)^2 \bmod n$$

- Če je e **liho**:

$$a^e \bmod n = \left[a \cdot (a^{e-1} \bmod n)\right] \bmod n$$

Algoritem hitrega potenciranja

1. Nastavi $result = 1$
2. Ponovi dokler $e > 0$:
   1. Če je e liho: $result = (result \cdot m) \bmod n$
   2. $e = \lfloor e/2 \rfloor$
   3. $m = (m \cdot m) \bmod n$
3. Vrni *result*

Primer izračuna

Naj bo:

$$\mathrm{m} = 4, \quad \mathrm{e} = 13, \quad \mathrm{n} = 497$$

Koraki:

$$\begin{aligned}
13 \text{ (liho)}: &\quad \mathit{result} = 1 \cdot 4 \bmod 497 = 4 \\
&\quad e = 6, \quad m = 4^2 \bmod 497 = 16 \\
6 \text{ (sodo)}: &\quad e = 3, \quad m = 16^2 \bmod 497 = 256 \\
3 \text{ (liho)}: &\quad \mathit{result} = 4 \cdot 256 \bmod 497 = 30 \\
&\quad e = 1, \quad m = 256^2 \bmod 497 = 429 \\
1 \text{ (liho)}: &\quad \mathit{result} = 30 \cdot 429 \bmod 497 = 445
\end{aligned}$$

Končni rezultat: $4^{13} \bmod 497 = 445$

Šifriranje in dešifriranje I

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
// Hitro potenciranje z modulom: (base^exp) mod mod
long long modPow(long long base, long long exp, long long mod) {
    long long result = 1;
    base %= mod;
    while (exp > 0) {
        if (exp % 2 == 1)
            result = (result * base) % mod;
        exp /= 2;
        base = (base * base) % mod;
    }
    return result;
}
int main(int argc, char* argv[]) {
    if (argc != 6) {
        cerr << "Uporaba: " << argv[0]
        << " enc|dec n e/d vhodna_datoteka izhodna_datoteka\n";
        return 1;
    }
    string mode = argv[1];
    long long n = stoll(argv[2]);
    long long key = stoll(argv[3]);
    string inputFile = argv[4];
    string outputFile = argv[5];
```

Šifriranje in dešifriranje II

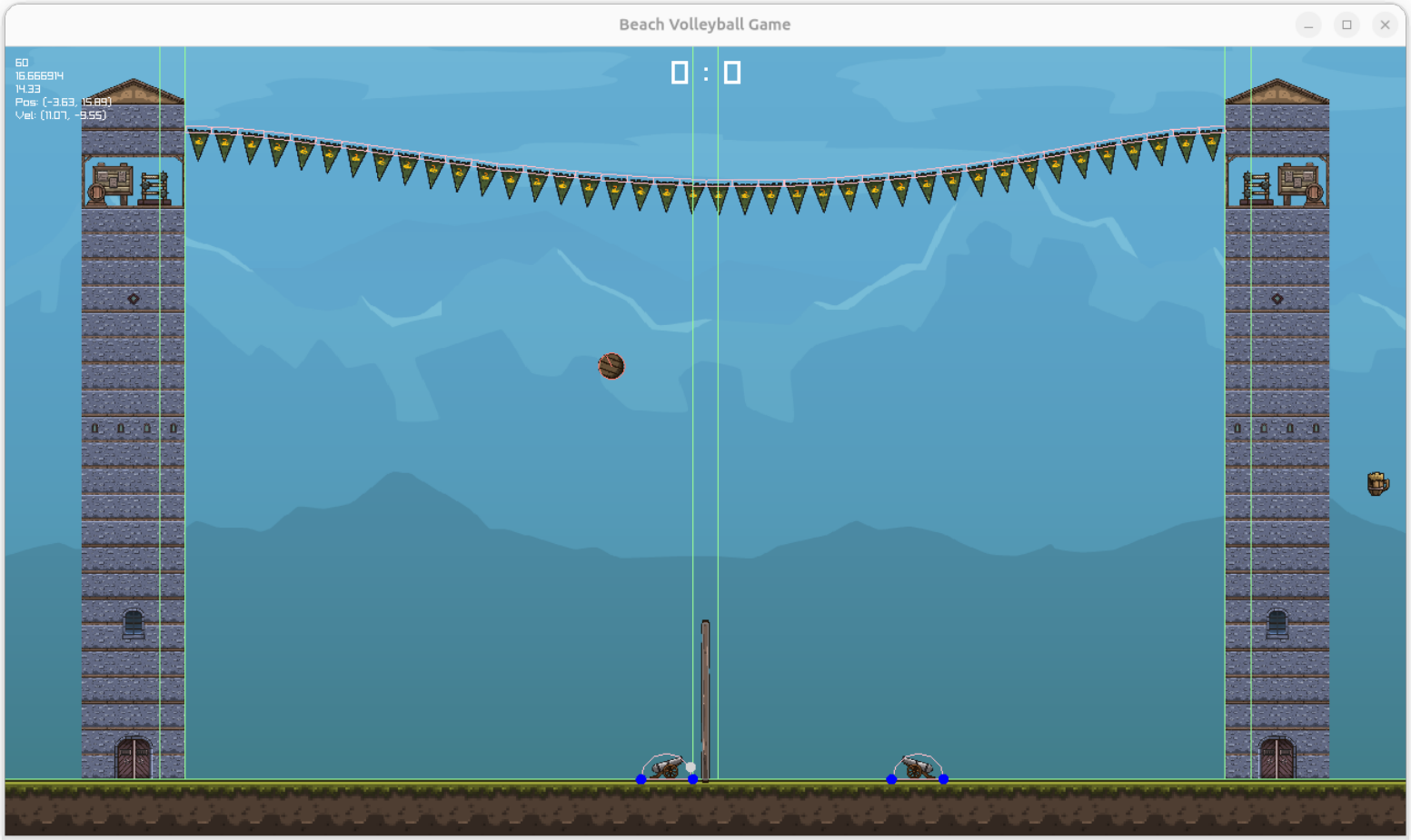
```
    ifstream in(inputFile, ios::binary);
    if (!in) {
        cerr << "Napaka pri odpiranju vhodne datoteke.\n";
        return 1;
    }
    ofstream out(outputFile, ios::binary);
    if (!out) {
        cerr << "Napaka pri odpiranju izhodne datoteke.\n";
        return 1;
    }
    if (mode == "enc") { // Sifriranje
        char byte;
        while (in.get(byte)) {
            unsigned char ubyte = static_cast<unsigned char>(byte);
            long long encrypted = modPow((long long)ubyte, key, n);
            out.write(reinterpret_cast<const char*>(&encrypted),
                      sizeof(encrypted));
        }
        cout << "Datoteka uspesno zakodirana.\n";
    }
    else if (mode == "dec") { // Desifriranje
        long long encrypted;
        while (in.read(reinterpret_cast<char*>(&encrypted), sizeof(encrypted))) {
            long long decrypted = modPow(encrypted, key, n);
            unsigned char byte = static_cast<unsigned char>(decrypted);
            out.put((char)byte);
        }
        cout << "Datoteka uspesno dekodirana.\n";
```

Šifriranje in dešifriranje III

```
    }
    else {
        cerr << "Neznan nacin dela: " << mode << ". Uporabi 'enc' ali 'dec'.\n";
        return 1;
    }
    return 0;
}
```

Igra odbojke

Odbojka


Beach Volleyball Game
60
16.666914
14.33
Pos: (-3.63, 15.89)
Vel: (11.07, -9.55)
0 : 0

Opis

- C++
- Windows, Linux in MacOS
- Box2d (`github.com/erincatto/box2d`)
- Raylib (`github.com/raysan5/raylib`)
- Igra brez igralcev (*angl. zero player game*)

```
// Debug risanje (debug draw)
#define BVOLLEY_PHYSICS_DEBUG_DRAW 1
#define BVOLLEY_DRAW_STATS 1
```

Igra brez igralcev

- Mere igrišča: 0.0 do 20.0
- Največ 3-je dotiki žoge
- Kontroler (*angl. controller*) - funkcija, ki izbere naslednjo akcijo, glede na trenutno stanje

```
struct Info {
  float player_x;
  float player_y;
  float ball_x;
  float ball_y;
  float ball_vel_x;
  float ball_vel_y;
};

struct Action {
  bool jump;
  float move_to_x;
};
```

Kontroler

- Implementacija svojega kontrolerja
- Tekomvanje med implementacijami

```
Action dummy_shooter_controller(const Info& info) {
  const auto dist_to_ball = static_cast<float>(
        std::sqrt(std::pow(info.ball_x - info.player_x, 2) +
        std::pow(info.ball_y - info.player_y, 2))
    );

  return Action{
      .jump =
        dist_to_ball < 2.0f,
      .move_to_x =
        (info.ball_x > 0.0f) ? info.ball_x : 10.0f,
  };
}
```

Sledenje žogi

- Kamera počasi zasleduje pozicijo žoge

```
const Vector2 ball_position = court.get_ball_position();

camera.target.x +=
  (ball_position.x - camera.target.x) * 0.01f;

camera.target.x =
    std::clamp(camera.target.x, -50.0f, 50.0f);
```