

# MASZYNA TURINGA

---

## Opis programu

Program jest symulacją maszyny Turinga, która służy do wykonywania algorytmów. Użytkownik wprowadza rozkazy, które zostają zapisane na nieskończenie długiej taśmie. Rozkazów może być nieskończona ilość, mogą one również być wczytywane z pliku z rozszerzeniem „\*.tur” lub już gotowa maszyna być zapisana do pliku. Rozkazy wprowadzane przez użytkownika mają jednak pewne ograniczenia. Stany, w których maszyna będzie się znajdowała mogą być łańcuchami znaków o dowolnej długości, jednak odczytywane symbol (z taśmy), zapisywane (na taśmie) oraz kierunek ruchu głowicy maszyny są ograniczone do jednego znaku. Maszyna może poruszać się w dwóch kierunkach, lewą lub prawą. Program wymaga zainstalowania biblioteki VISUALA oraz biblioteki FLTK. Program działa pod systemami Windows i Linux.

## Instrukcja obsługi

Program uruchamia się poprzez dwukrotne kliknięcie myszy na COŚ.exe. W momencie uruchomienia programu uruchamia się prosty i przejrzysty interfejs graficzny, gdzie każda funkcjonalność jest odpowiednio nazwana, lub dodatkowo jest opisana. Jeżeli chcemy wczytać listę rozkazów do naszej maszyny, powinniśmy wybrać opcję Program->Otwórz (lub nacisnąć kombinację klawiszy Ctrl+ o). Następnie należy wybrać plik z listy i nacisnąć na przycisk Otwórz. Maszyna automatycznie się załaduje. Maszynę można również zapisać, klikając Program->Zapisz (Ctrl+ s). Pojawi się okienko, w którym należy wybrać ścieżkę zapisu pliku, oraz w odpowiednim miejscu wpisać nazwę pliku. Po naciśnięciu przycisku „Zapisz” maszyna zostanie zapisana. Aby wprowadzić ręcznie kolejne rozkazy, w głównym oknie należy nacisnąć przycisk „Dodaj”, który otworzy okno potrzebne do wprowadzenia kolejnych elementów rozkazu. Każde pole jest odpowiednio podpisane oraz wyjaśnione. Po zakończeniu wprowadzania rozkazu należy nacisnąć przycisk „Dodaj”, co zakończy procedurę dodawania kolejnego rozkazu. Gdy naciśniemy w przycisk „Start”, maszyna rozpocznie wykonywanie rozkazów. W dowolnym momencie możemy rozpocząć pracę z nową maszyną, wystarczy wybrać opcję Program->Nowy program. Program też udostępnia informacje o aktualnej wersji i autorach projektu, dokonuje się tego poprzez wybranie opcji Opcje->Informacje o programie (Ctrl + i). Gdy chcemy zakończyć pracę programu, wybieramy opcję Program->Zakończ.


## Specyfikacja techniczna

Projekt składa się z 8 plików nagłówkowych oraz 9 plików źródłowych. Pliki nagłówkowe zawierają definicje klas. Plik application.hpp zawiera definicję klasy Application, która odpowiada za interfejs graficzny aplikacji, bezpośrednią operację zapisu i otworzenia maszyny z plików oraz operację stworzenia nowej maszyny. Plik machine.hpp definiuje silnik maszyny

turinga. Plik menu.hpp odpowiada za obsługę menu, w tym rozpoczęcie operacji otworzenia nowej maszyny, otworzenia istniejącej lub zapisu maszyny, zakończenia pracy programu czy pokazania informacji o programie. Nagłówek popup.hpp obsługuje okienko dodania nowego rozkazu. Plik state\_map.hpp zawiera w sobie strukturę Move, która jest odpowiedzialna za przetrzymywanie symbolu wpisanego na taśmę, kierunku ruchu głowicy oraz stanu, w jakim się maszyna znajdzie po wykonaniu rozkazu. Klasa SymMap i StateMap służą do przechowywania programu maszyny Turinga. Nagłówek states\_table.hpp obsługuje tabelę stanów (wraz z rozkazami), które znajdują się w klasie StateMap. Pliki o tych samych nazwach z rozszerzeniem .cpp są analogiczne i zawierają ciała metod, konstruktorów itd. Plik controller.hpp definiuje klasę kontrolera łączącą wszystkie elementy aplikacji by razem działały. Plik tape.hpp definiuje klasę odpowiedzialną za wyświetlanie taśmy maszyny. Plik main.cpp rozpoczyna pracę programu. Aby skompilować poprawnie program, należy dołączyć bibliotekę FLTK oraz biblioteki VISUAL STUDIO i podłączyć wszystkie pliki, które zostały do tej pory wymienione.

## Szczegóły techniczne

Projekt korzysta z języka C++ w wersji 11. Do kontroli pamięci używana jest klasa `std::unique_ptr` np:

```
 class Popup : public Fl_Window {  
    public:  
        typedef std::unique_ptr<Fl_Box> BoxPtr;  
        typedef std::unique_ptr<Fl_Input> InputPtr;  
        typedef std::unique_ptr<Fl_Button> ButtonPtr;  
  
    [...]  
        InputPtr state1_;  
    [...]  
        BoxPtr state1_text_;  
    [...]  
        ButtonPtr btn_;  
};  
  
Popup::Popup(Controller* ctrl):  
    Fl_Window(300, 400, "Dodaj rozkaz"), ctrl_(ctrl) {  
  
        state1_ = InputPtr(new Fl_Input(110, 30, 180, 17, "Stan początkowy: "));  
    [...]  
        sym2_text_ = BoxPtr(new Fl_Box(10,175,280,50,"Symbol, który będzie zapisany do  
tasmy (np.: '1')."));  
    [...]  
        btn_ = ButtonPtr(new Fl_Button(230, 350, 50, 30, "Dodaj"));  
    [...]  
        end();  
}
```

Projekt korzysta z biblioteki FLTK w wersji 1.3. Do tabeli stanów wykorzystywana jest klasa `Fl_Table_Row`, która rysuje komórki w następujący sposób:

```

void StatesTable::draw_cell(TableContext context, int R, int C, int X, int Y, int W, int H) {

    switch (context) {
    case CONTEXT_STARTPAGE:
        fl_font(FL_TIMES, 12);
        return;
    case CONTEXT_ROW_HEADER:
    case CONTEXT_COL_HEADER:
        fl_push_clip(X, Y, W, H);

        fl_draw_box(FL_THIN_UP_BOX, X, Y, W, H, color());
        fl_color(FL_BLACK);
        fl_draw(headers[C].c_str(), X, Y, W, H, FL_ALIGN_CENTER);

        fl_pop_clip();
        break;

    case CONTEXT_CELL:
        fl_push_clip(X, Y, W, H);

        fl_color( FL_BACKGROUND_COLOR);
        fl_rectf(X, Y, W, H);

        fl_color(FL_BLACK);

        fl_draw_box(FL_THIN_UP_BOX, X, Y, W, H, color());
        fl_color(FL_BLACK);
        fl_draw(table_[R][C].c_str(), X, Y, W, H, FL_ALIGN_CENTER);

        fl_pop_clip();
        break;
    default:
        break;
    }
}

```

## Podział pracy

### Michał Muskała:

- opracowanie zasad działania maszyny
- klasa Machine
- klasa Controller
- klasy StateMap, SymMap i Move
- klasa StatesTable
- architektura programu i podział kodu na jednostki kontoli

**Kamil Kryus:**

- opracowanie interfejsu użytkownika i zasad interakcji z aplikacją.
- klasa Application
- klasa Menu
- klasa Popup