Symulacja Mrówki Langtona

Program został stworzony przez: Rybachok Andrii, Chaienko Oleksandr

GitHub: https://github.com/andrii-rybachok/LangtonAnt-JIMP-2023/tree/main

Opis Programu

Program symuluje ruch mrówki Langtona na planszy. Mrówka Langtona jest abstrakcyjnym automatem komórkowym, który porusza się na dwuwymiarowej siatce. Kierunek ruchu mrówki zmienia się w zależności od koloru komórki, na której się znajduje.

Sposób Użycia

Aby skorzystać z programu, należy uruchomić go z linii poleceń, podając odpowiednie parametry:

- -r: Liczba wierszy planszy.
- -c: Liczba kolumn planszy.
- -i: Liczba iteracji symulacji.
- -d: Kierunek początkowy mrówki (0 - dół, 1 - lewo, 2 - góra, 3 - prawo).
- -b: Procent czarnych komórek na planszy.
- -p: Prefix nazw plików wyjściowych.
- -m: Nazwa pliku dla wyczytania początkowej mapy(opcjonalnie).
- -a: Służy dla animacji mrówki w terminale.

Idee Działania Mrówki Langtona

Mrówka chodzi po dwuwymiarowej siatce. Może poruszać się w jednym z 4-ech kierunków (góra, dół, lewo, prawo), zgodnie z następującymi zasadami:

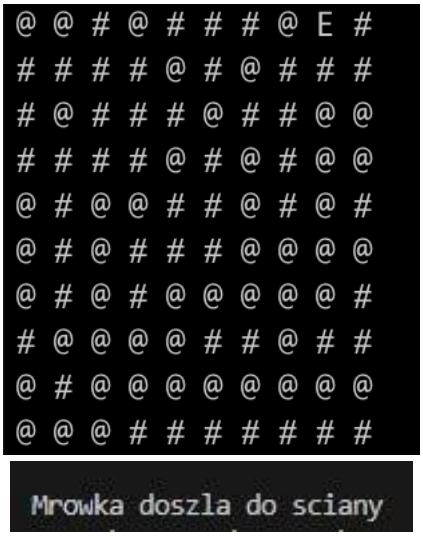
- Mrówka znajduje się w komórce białej, wykonuje: obrót o 90 stopni w prawo, zmienia kolor komórki na przeciwny, przesuwa się o jedną komórkę do przodu
- Mrówka znajduje się w komórce czarnej, wykonuje: obrót o 90 stopni w lewo, zmienia kolor komórki na przeciwny, przesuwa się o jedną komórkę do przodu.
- Znaki mrówki jaki korzystamy dla programy:

```
#define ARROW_NORTH_WHITE '^'
#define ARROW_NORTH_BLACK 'N'
#define ARROW_EAST_WHITE '>'
#define ARROW_EAST_BLACK 'E'
#define ARROW_SOUTH_WHITE 'v'
#define ARROW_SOUTH_BLACK 'S'
#define ARROW_WEST_WHITE '<'
#define ARROW_WEST_BLACK 'W'
#define SQUARE_WHITE '#'
#define SQUARE_BLACK '@'
```

Rys 1. Znaki

- Ściana (koniec planszy):

Jeśli mrówka dotrze do końca planszy, pozostanie na miejscu, a program powiadomi użytkownika, że mrówka dotarła do końca planszy.



Rys 2. Mrówka dotarła końca planszy

Podział Programu na Moduły

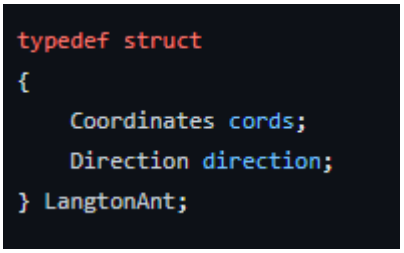
Program został podzielony na moduły w celu lepszej organizacji kodu i zwiększenia czytelności. Moduły to:

- **langtonAnt.h**: Zawiera deklaracje struktury **LangtonAnt.c** oraz funkcji związanych z mrówką Langtona.
- **coordinates.h**: Deklaracja struktury **Coordinates,c** używanej do reprezentowania współrzędnych.
- **cellColor.h**: Deklaracja enumeracji **CellColor.c** reprezentującej kolory komórek (białe, czarne).
- **langtonField.h**: Zawiera deklaracje struktury **LangtonField.c** oraz funkcji do obsługi planszy.
- **main.c**: Główny plik programu zawierający funkcję i logikę obsługi argumentów oraz symulacji.

Opis podstawowych funkcji i struktur

1. Struktury:

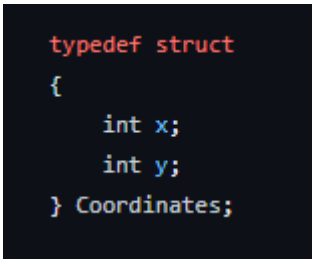
- LangtonAnt (langtonAnt.h):



Rys 3. Koordynaty i kierunek mrówki

1) Struktura reprezentująca mrówkę Langtona.
2) Zawiera współrzędne (cords) i kierunek (direction) mrówki.

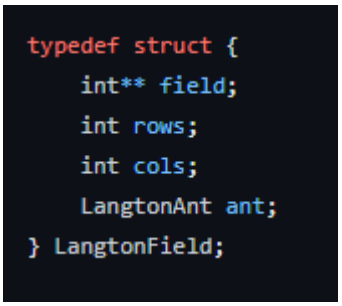
- Coordinates (coordinates.h):



Rys 4. Koordynaty na plansze

1) Struktura reprezentująca współrzędne na planszy.

- LangtonField (langtonField.h):

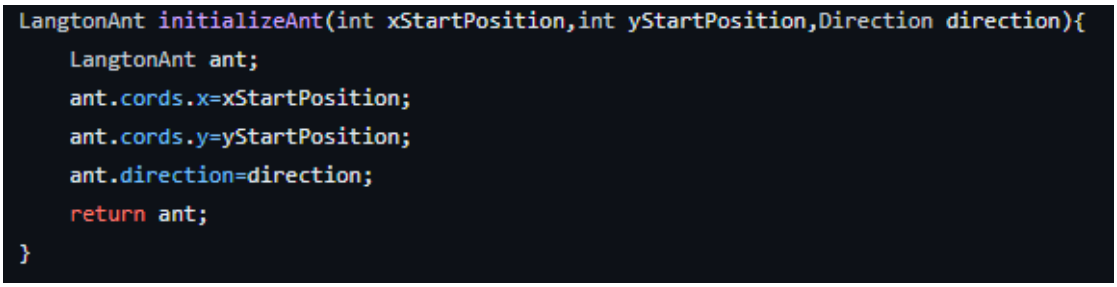


Rys 4. Plansza z morówką

1) Struktura reprezentująca planszę i mrówkę Langtona.
2) Zawiera dwuwymiarową tablicę (field), liczby wierszy i kolumn (rows i cols) oraz mrówkę.

2. Funkcje:

- initializeAnt (langtonAnt.h):



Rys 5. Inicjalizacja mrówki

Inicjalizuje strukturę LangtonAnt z określonymi współrzędnymi startowymi i kierunkiem.

- antIterate (langtonAnt.h):

```c
int antIterate(LangtonAnt* ant,CellColor color,int xMax,int yMax){
    switch (color)
    {
        case Black:
            ant->direction = ToLeftChangeDirection(ant->direction);
            break;
        case White:
            ant->direction = ToRightChangeDirection(ant->direction);
            break;
    default:
        break;
    }
    move(&(ant->cords),ant->direction);


    if((ant->cords.x>xMax || ant->cords.x<0 || ant->cords.y>yMax || ant->cords.y<0)){
        return -1;
    }
    return 0;
}
```

Rys 6. Realizacja ruchu mrówki na komórkach

Realizuje jedną iterację ruchu mrówki w zależności od koloru aktualnej komórki. Aktualizuje współrzędne mrówki i zwraca -1, jeśli mrówka opuściła planszę.

- toggleColor (cellColor.h):

```c
#include "cellColor.h"
CellColor toggleColor(CellColor color){
    switch (color)
    {
        case White:
            return Black;
        case Black:
            return White;
    default:
        break;
    }
    return color;
}
```

Rys 7. Zmiana komórek

Zmienia kolor komórki (biały na czarny, czarny na biały).

- initializeField (langtonField.h):

```
LangtonField initializeField(int rows, int cols, int blackColsPercent, Direction antStartDirection) {
    LangtonField lngField;
    lngField.cols = cols;
    lngField.rows = rows;

    lngField.field = malloc(rows * sizeof(int*));
    for (int i = 0; i < rows; i++) {
        lngField.field[i] = calloc(cols, sizeof(int));
    }

    if (lngField.field == NULL) {
        fprintf(stderr, "Allocation memory error for field!\n");
        exit(EXIT_FAILURE);
    }

    int countOfBlackCols = (int)((rows * cols) * (blackColsPercent / 100.0));
    srand(time(NULL));

    for (int i = 0; i < countOfBlackCols; i++) {
        int rndRow = rand() % rows;
        int rndCol = rand() % cols;

        if (lngField.field[rndRow][rndCol] == 0) {
            lngField.field[rndRow][rndCol] = 1;
        } else {
            i--;
        }
    }

    lngField.ant = initializeAnt((int)ceil((double)cols / 2) - 1, (int)ceil((double)rows / 2) - 1, antStartDirection);

    return lngField;
}
```

Rys 8. Inicjalizacja planszy

Inicjalizuje strukturę LangtonField z określoną liczbą wierszy, kolumn, procentem czarnych komórek i kierunkiem startu mrówki.

- fieldIterate (langtonField.h):

```
int fieldIterate(LangtonField* langField){
    if(langField->field==NULL){
        fprintf(stderr, "Field does not initialized!\n");
        exit(EXIT_FAILURE);
    }

    int* cell = &(langField->field[langField->ant.cords.y][langField->ant.cords.x]);

    CellColor color= *cell;
    int returnValue= antIterate(&langField->ant,color,langField->cols-1,langField->rows-1);
    color = toggleColor(color);
    *cell = (int)color;
    return returnValue;
}
```

Rys 9. Realizacja ruchu mrówki na plansze

Realizuje jedną iterację ruchu mrówki na planszy.

Aktualizuje stan planszy i zwraca -1, jeśli mrówka opuściła planszę.

- printField (langtonField.h):

```
void printField(LangtonField* field) {
    for (int i = 0; i < field->rows; i++) {
        for (int j = 0; j < field->cols; j++) {
            if (field->ant.cords.x == j && field->ant.cords.y == i) {
                switch (field->ant.direction) {
                    case Top:
                        printf("^[%d] ", field->field[i][j]);
                        break;
                    case Bottom:
                        printf("b[%d] ", field->field[i][j]);
                        break;
                    case Left:
                        printf("<[%d] ", field->field[i][j]);
                        break;
                    case Right:
                        printf(">[%d] ", field->field[i][j]);
                        break;
                    default:
                        break;
                }
            } else {
                printf("%d ", field->field[i][j]);
            }
        }
        printf("\n");
    }
}
```

Rys 10. Wpisuje stan planszy

Wypisuje stan planszy na standardowe wyjście.

- printFieldToFile (langtonField.h):

```
void printFieldToFile(LangtonField* field, FILE* file) {
    for (int i = 0; i < field->rows; i++) {
        for (int j = 0; j < field->cols; j++) {
            if (field->ant.cords.x == j && field->ant.cords.y == i) {
                char antSymbol;
                switch (field->ant.direction) {
                    case Top:
                        antSymbol = (field->field[i][j] == 0) ? ARROW_NORTH_WHITE : ARROW_NORTH_BLACK;
                        break;
                    case Bottom:
                        antSymbol = (field->field[i][j] == 0) ? ARROW_SOUTH_WHITE : ARROW_SOUTH_BLACK;
                        break;
                    case Left:
                        antSymbol = (field->field[i][j] == 0) ? ARROW_WEST_WHITE  : ARROW_WEST_BLACK ;
                        break;
                    case Right:
                        antSymbol = (field->field[i][j] == 0) ? ARROW_EAST_WHITE : ARROW_EAST_BLACK;
                        break;
                    default:
                        break;
                }
                fprintf(file, "%c", antSymbol);
            } else {
                char squareSymbol = (field->field[i][j] == 0) ? SQUARE_WHITE : SQUARE_BLACK;
                fprintf(file, "%c", squareSymbol);
            }
            fprintf(file, " ");
        }
        fprintf(file, "\n");
    }
}
```

Rys 11. Zapis do pliku

Zapisuje stan planszy do pliku.

- loadMapFromFile (langtonField.h):

```
void loadMapFromFile(LangtonField* langField,const char* mapFileName) {
    char line[256];
    int row =0;
    FILE* file = fopen(mapFileName, "r");
    if (file == NULL) {

        perror("Error opening file");
        exit(EXIT_FAILURE);
    }
    while (fgets(line, sizeof(line), file)) {
        int j=0;
        for (int i = 0; i < strlen(line); i++)
        {
            if(line[i]!=' ' && line[i]!='\n'){
                switch (line[i])
                {
                    case SQUARE_WHITE:
                        langField->field[row][j]=0;
                        break;
                    case SQUARE_BLACK:
                        langField->field[row][j]=1;
                        break;
                    case ARROW_NORTH_WHITE:
                    case ARROW_SOUTH_WHITE:
                    case ARROW_WEST_WHITE:
                    case ARROW_EAST_WHITE:
                        langField->field[row][j]=0;
                        langField->ant = initializeAnt(j,row,getAntDirection(line[i]));
                        break;
                    case ARROW_NORTH_BLACK:
                    case ARROW_SOUTH_BLACK:
                    case ARROW_WEST_BLACK:
                    case ARROW_EAST_BLACK:
                        langField->field[row][j]=1;
                        langField->ant = initializeAnt(j,row,getAntDirection(line[i]));
                        break;
                    default:
                        break;
                }
                j++;
            }
        }
        row++;
    }
    fclose(file);
}
```

Rys 12. Wczytanie mapy z pliku

Wczytuje mapę planszy z pliku do struktury LangtonField.

- initializeFieldWithMap (langtonField.h):

```c
LangtonField initializeFieldWithMap(int rows, int cols,  const char* mapFileName) {
    FILE* file = fopen(mapFileName, "r");

    if (file == NULL) {

        perror("Error opening file");
        exit(EXIT_FAILURE);
    }
    LangtonField lngField;
    checkMap(file,&lngField.cols,&lngField.rows);

    lngField.field = malloc(rows * sizeof(int*));
    for (int i = 0; i < rows; i++) {
        lngField.field[i] = calloc(cols, sizeof(int));
    }

    if (lngField.field == NULL) {
        fprintf(stderr, "Allocation memory error for field!\n");
        exit(EXIT_FAILURE);
    }

    loadMapFromFile(&lngField, mapFileName);

    fclose(file);
    return lngField;
}
#define BUF_SIZE 65536
```

Rys 13. Inicjalizacja mapy z pliku

Inicjalizuje planszę z mapy wczytanej z pliku.

- checkMap (langtonField.h):

```c
int checkMap(FILE* file,int* cols,int* rows)
{
    char buf[BUF_SIZE];
    *rows = 0;
    *cols=0;

    while(fgets(buf, BUF_SIZE, file)) {
        if(strlen(buf)>1){
            *rows+=1;
            if(*rows==1){
                for(int i = 0; i < strlen(buf); i++){
                    if(buf[i]!=' ' && buf[i]!='\n'){
                        *cols+=1;
                    }
                }
            }
        }
    }

    return 0;
}
```

Rys 14. Analiz mapy z pliku

służy do analizy zawartości pliku.

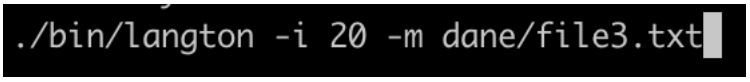
Przykładowe Działanie Programu

Uruchomienie programu z przykładowymi parametrami:

Rys 15, Przykład działania

Program utworzy pliki wynikowe z symulacją mrówki Langtona na planszy 10x10 i z 50% czarnymi komórkami, wykonując 12 iteracji i pokazuje animacji w terminale. Działanie programu można zobaczyć w plikach wyjściowych o nazwach "file_nriteracji**n**.txt", gdzie **n** to numer iteracji.

Uruchomienie programu z przykładowymi parametrami(opcjonalnie):



Rys 16, Uruchomienie mapy z pliku

Program utworzy pliki wynikowe z symulacją mrówki Langtona zgodnie z parametrami jaki jest w pliku (kolumny, wiersze i procent czarnych komórek) Działanie programu można zobaczyć w plikach wyjściowych o nazwach "file_nriteracji**n**.txt", gdzie **n** to numer iteracji.

Wnioski

Program umożliwia eksperymentowanie z różnymi ustawieniami symulacji mrówki Langtona. Działanie programu jest zgodne z zasadami ”projekt23Z”, a wyniki są zależne od początkowych warunków i reguł ruchu mrówki.