# Programowanie III
# dokumentacja projektu Statki

Michał Pluszczewski grupa 4G
Politechnika Śląska
Wydział: Matematyki Stosowanej
Kierunek: Informatyka

24 stycznia 2022

# Część I

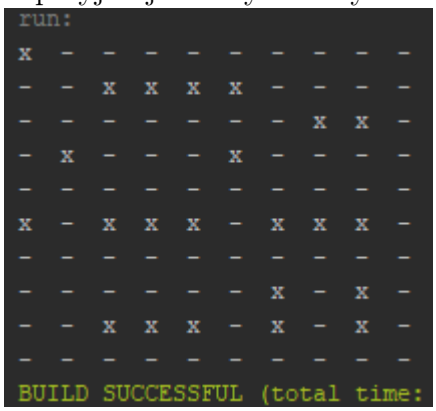
## Opis programu Statki

W grze w statki należy przygotować planszę wielkości 10 na 10 kratek, na której rozmieszczone mają być: cztery jednomasztowce (jedna kratka), trzy dwumasztowce (dwie kratki), trzy trójmasztowce (trzy kratki) i jeden czteromasztowiec (cztery kratki). Zasady rozmieszczania statków na planszy są następujące:
a) pola dwumasztowców, trójmasztowców i czteromasztowca muszą być tak umieszczone na planszy, aby z każdego pola danego statku dało się przejść do każdego innego pola tego statku przechodząc wyłącznie przez ich wspólne boki;
b) dwa różne statki nie mogą się stykać ze sobą ani bokami, ani wierzchołkami. Napisz program, który zwracał będzie losowe i poprawne rozmieszczenie statków na planszy. Plansza ma być wyświetlana ma monitorze, a statki zaznaczone mają być na niej znakami „x".

# Część II

## Opis działania

Program nie przyjmuje żadnych danych wejściowych.




Rysunek 1: Program po uruchomieniu

Program wypisuje w losowym punkcie na planszy w konsoli cztery jednomasztowce (jedna kratka), trzy dwumasztowce (dwie kratki), trzy trójmasztowce (trzy kratki) i jeden czteromasztowiec (cztery kratki), które nie mogą się ze sobą stykać ani bokami, ani wierzchołkami.

## Algorytm

Program jest podzielony na kilka funkcji, umieszczonych w klasie Statki.

W funkcji wypelnienie planszy wypełniamy naszą planszę pustymi znakami -.

W funkcji wypisanie planszy wypisujemy planszę na ekran.

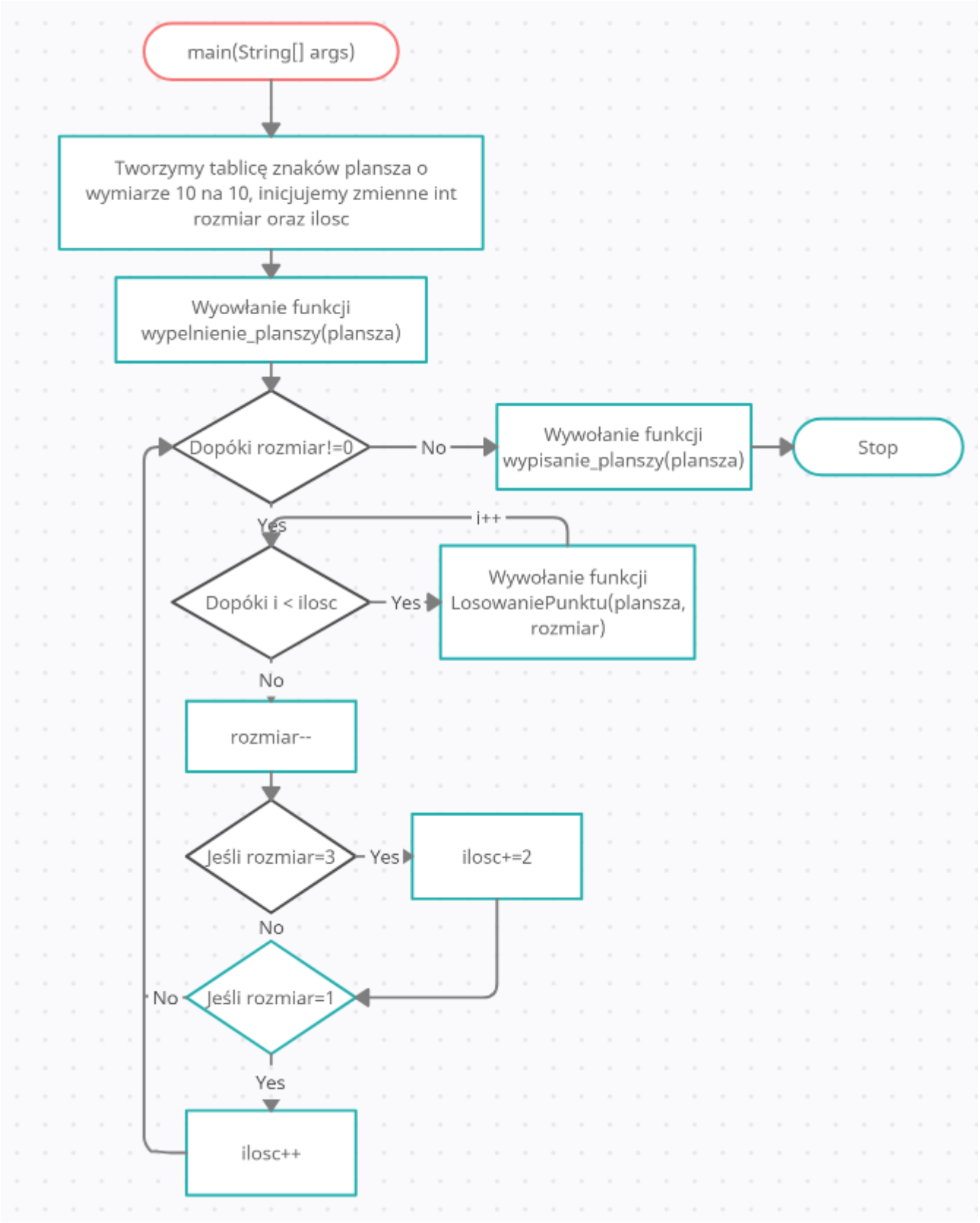
W funkcji getRandomNumber losujemy losową liczbę z danego przedziału.

W funkcji LosowaniePunktu losujemy dwie liczby od 0 do 9 oraz kierunek w którym wypiszemy dany masztowiec (liczby od 0 do 3). Następnie wywołujemy funkcję sprawdz punkt i jeśli funkcja ta zwróci wartość true przechodzimy do wypisywania naszego masztowca w odpowiednim miejscu na planszy.

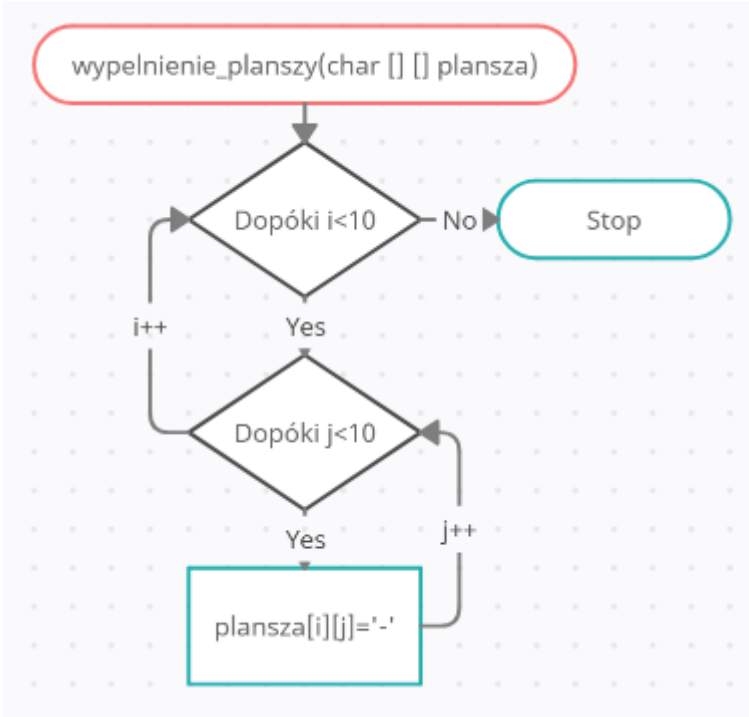
W funkcji sprawdz punkty sprawdzamy czy możemy zapisać masztowiec na planszy(jego otoczenie jak i miejsce w którym chcemy go zapisać). Jeśli choć jedno sprawdzane pole jest różne od znaku - funkcja zwraca wartość false. Natomiast jeśli wszystkie warunki są spełnione zwaraca wartość true.

W funkcji main tworzymy plansze znaków 10 na 10, następnie wypełniamy planszę poprzez wysowałnie odpowiedniej funkcji, po czym odpowiednio dla rozmiaru i ilości masztowców losujemy i sprawdzamy punkty wywołując funkcję LosowaniePunktu. Wypisujemy planszę na ekran wywołując funkcję wypisanie planszy.

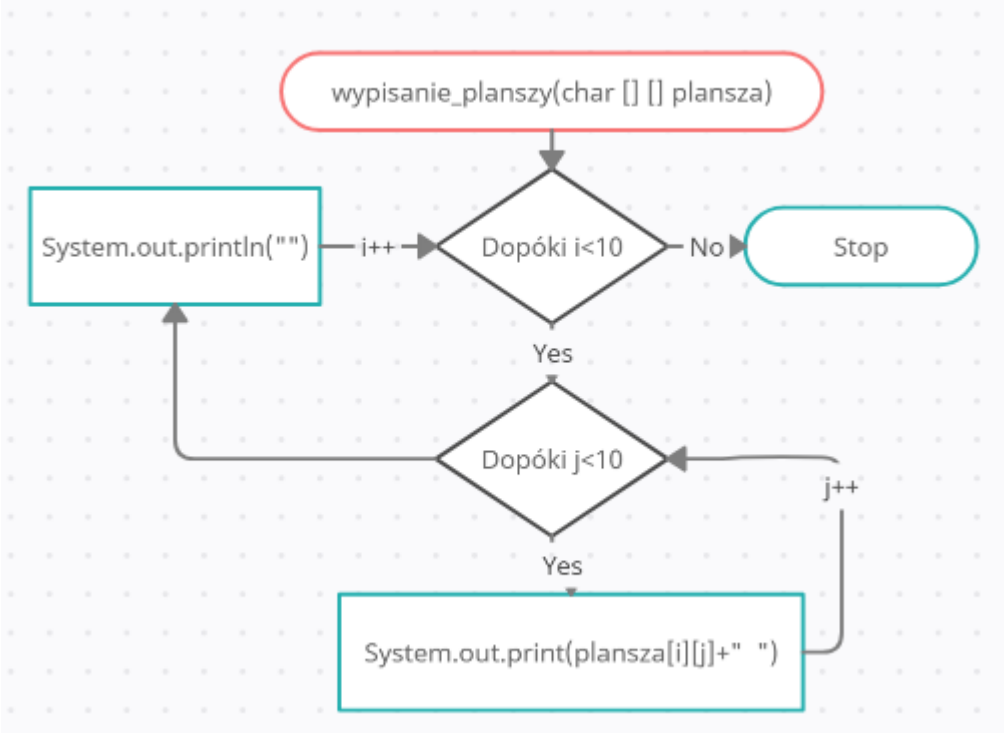
Schemat blokowy programu:



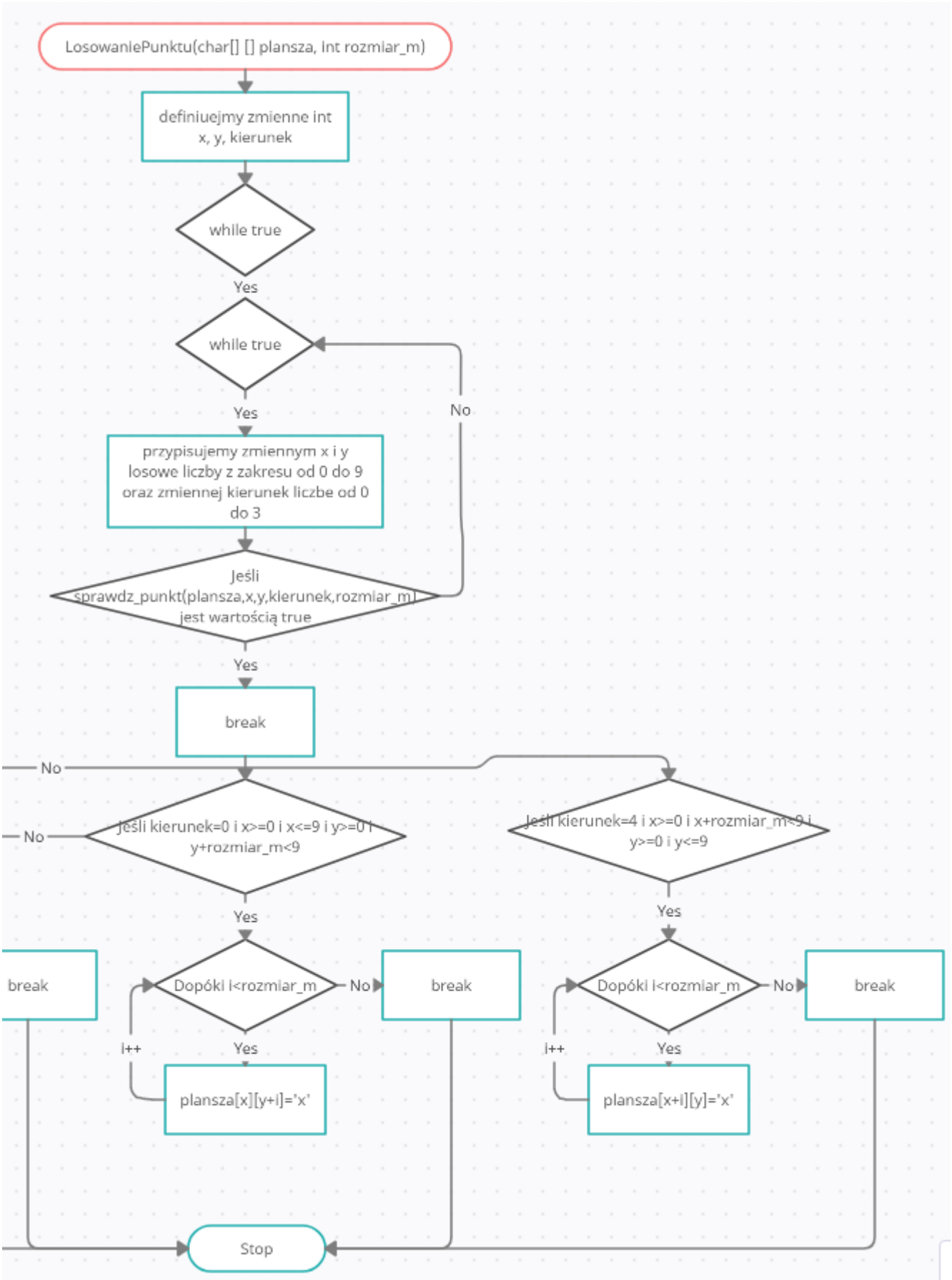

Rysunek 2: Schemat blokowy funkcji main

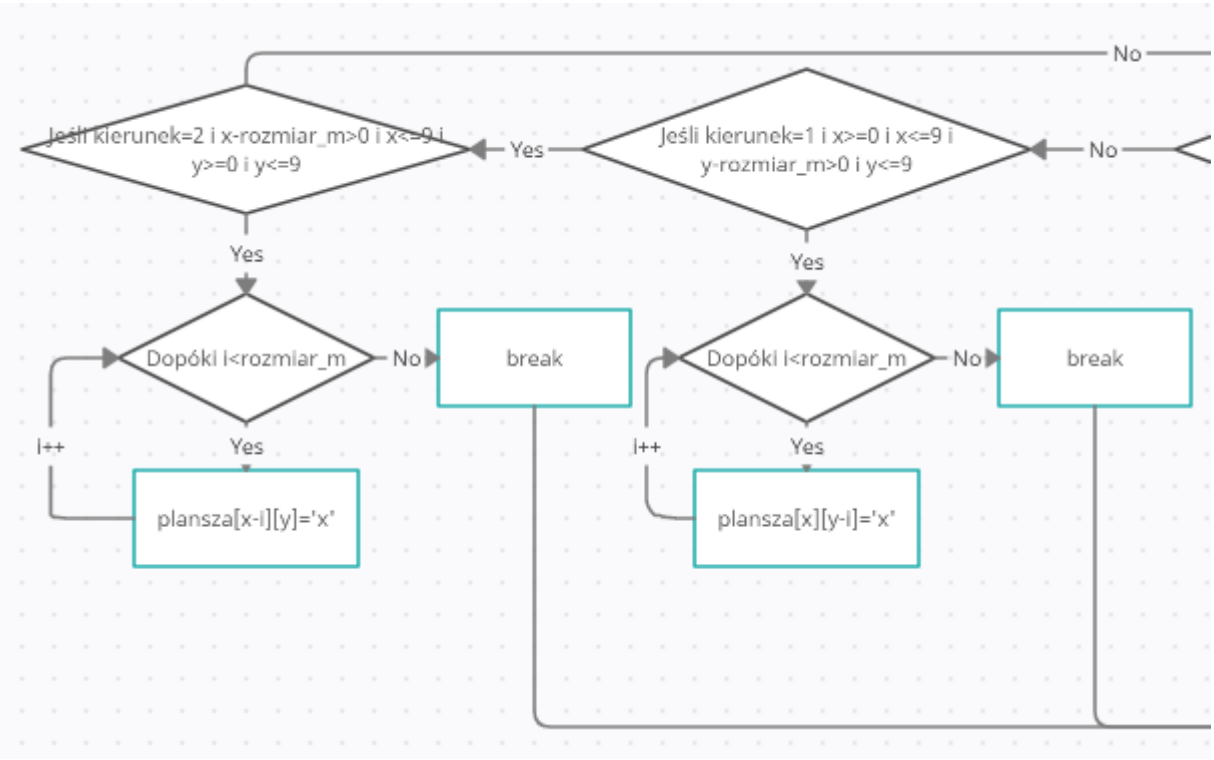



Rysunek 3: Schemat blokowy funkcji wypelnienie planszy

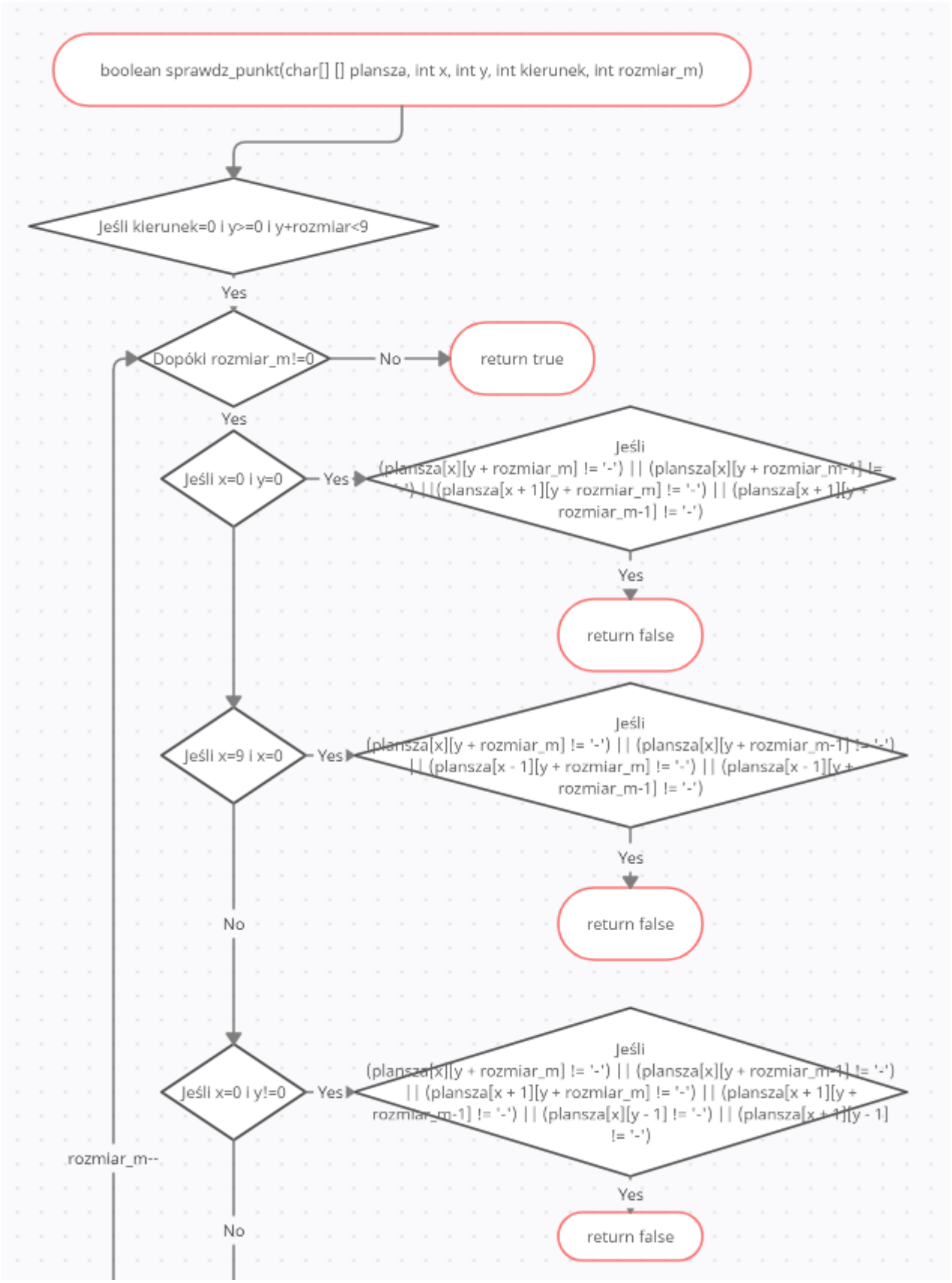


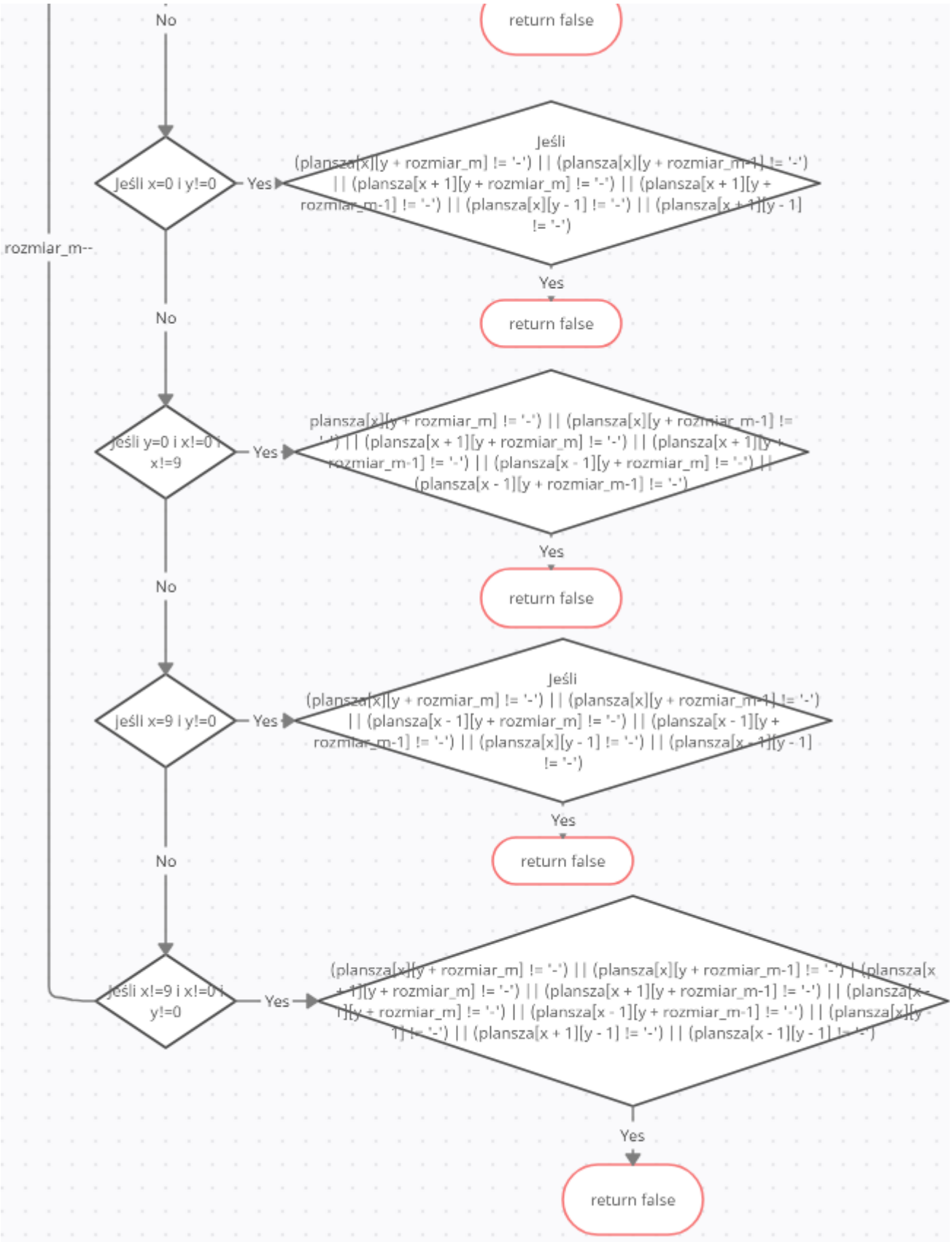

Rysunek 4: Schemat blokowy funkcji wypisanie planszy


LosowaniePunktu(char[] [] plansza, int rozmiar_m)
definiuejmy zmienne int x, y, kierunek
while true
Yes
while true
Yes
No
przypisujemy zmiennym x i y losowe liczby z zakresu od 0 do 9 oraz zmiennej kierunek liczbe od 0 do 3
Jeśli sprawdz_punkt(plansza,x,y,kierunek,rozmiar_m) jest wartością true
Yes
break
No
No
Jeśli kierunek=0 i x>=0 i x<=9 i y>=0 i y+rozmiar_m<9
Jeśli kierunek=4 i x>=0 i x+rozmiar_m<9 i y>=0 i y<=9
Yes
Yes
break
Dopóki i<rozmiar_m
No
break
Dopóki i<rozmiar_m
No
break
i++
Yes
i++
Yes
plansza[x][y+i]='x'
plansza[x+i][y]='x'
Stop




Rysunek 4: Schemat blokowy funkcji LosowaniePunktu


boolean sprawdz_punkt(char[] [] plansza, int x, int y, int kierunek, int rozmiar_m)
Jeśli kierunek=0 i y>=0 i y+rozmiar<9
Yes
Dopóki rozmiar_m!=0
No
return true
Yes
Jeśli x=0 i y=0
Yes
Jeśli
(plansza[x][y + rozmiar_m] != '-') || (plansza[x][y + rozmiar_m-1] != '-') ||(plansza[x + 1][y + rozmiar_m] != '-') || (plansza[x + 1][y + rozmiar_m-1] != '-')
Yes
return false
Jeśli x=9 i x=0
Yes
Jeśli
(plansza[x][y + rozmiar_m] != '-') || (plansza[x][y + rozmiar_m-1] != '-') || (plansza[x - 1][y + rozmiar_m] != '-') || (plansza[x - 1][y + rozmiar_m-1] != '-')
Yes
return false
No
Jeśli x=0 i y!=0
Yes
Jeśli
(plansza[x][y + rozmiar_m] != '-') || (plansza[x][y + rozmiar_m-1] != '-') || (plansza[x + 1][y + rozmiar_m] != '-') || (plansza[x + 1][y + rozmiar_m-1] != '-') || (plansza[x][y - 1] != '-') || (plansza[x + 1][y - 1] != '-')
Yes
return false
rozmiar_m--
No




Rysunek 4: Schemat blokowy funkcji sprawdz punkt

Sytuacja wygląda analogicznie dla innych kierunków

Pseudokod:

```
Data: Dane wejściowe: plansza znaków 10 na 10, rozmiar masztowca
while true do
    while true do
        x = getRandomNumber(0, 9);
        y = getRandomNumber(0, 9);
        kierunek = getRandomNumber(0, 3)
        if (sprawdzpunkt(plansza,x,y,kierunek,rozmiarm)==true) then
            break;
        end
    end
    if (kierunek==0 and (x >= 0 and x <= 9) and (y >= 0 and y+rozmiarm < 9))
     then
        for int i = 0; i < rozmiarm; i++ do
            plansza[x][y + i] = 'x';
        end
        break;
    end
    if (kierunek==1 and (x >= 0 and x <= 9) and (y−rozmiarm > 0 and y <= 9))
     then
        for int i = 0; i < rozmiarm; i++ do
            plansza[x][y − i] = 'x';
        end
        break;
    end
    if (kierunek==2 and (x−rozmiarm > 0 and x <= 9) and (y >= 0 and y <= 9))
     then
        for int i = 0; i < rozmiarm; i++ do
            plansza[x][y + i] = 'x';
        end
        break;
    end
    if (kierunek==3 and (x >= 0 and x+rozmiarm < 9) and (y >= 0 and y <= 9))
     then
        for int i = 0; i < rozmiarm; i++ do
            plansza[x][y + i] = 'x';
        end
        break;
    end
end
```

**Algorithm 1:** Algorytm losowania punktu.

## Implementacja

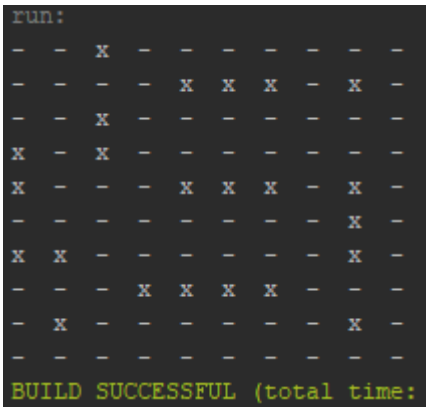
Implementacja pseudokodu napisanego w subsekcji Algorytm

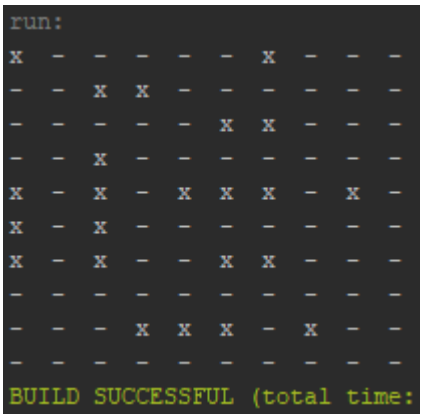
```
1 public static void LosowaniePunktu(char[][] plansza, int rozmiar_m){
2     int x, y, kierunek;
3     while(true){
4         while(true){
5             x=getRandomNumber(0,9);
6             y=getRandomNumber(0,9);
7             kierunek=getRandomNumber(0,3);
8             if (sprawdz_punkt(plansza,x,y,kierunek,rozmiar_m))
9                 break;
10        }
11       //prawo
12        if (kierunek == 0 && (x >= 0 && x <= 9) && (y >= 0 && y+
              rozmiar_m < 9)) {
13            for (int i = 0; i < rozmiar_m; i++) {
14                plansza[x][y + i] = 'x';
15            }
16            break;
17        }
18        //lewo
19        else if (kierunek == 1 && (x >= 0 && x <= 9) && (y-rozmiar_m > 0
              && y <= 9)) {
20            for (int i = 0; i < rozmiar_m; i++) {
21                plansza[x][y - i] = 'x';
22            }
23            break;
24        }
25        //gora
26        else if (kierunek == 2 && (x-rozmiar_m > 0  && x <= 9) && (y >=
              0 && y <= 9)) {
27            for (int i = 0; i < rozmiar_m; i++) {
28                plansza[x - i][y] = 'x';
29            }
30            break;
31        }
32        //dol
33        else if (kierunek == 3 && (x >= 0 && x+rozmiar_m < 9) && (y >= 0
              && y <= 9)) {
34            for (int i = 0; i < rozmiar_m; i++) {
35                plansza[x + i][y] = 'x';
36            }
37            break;
38        }
39    }
40 }
```

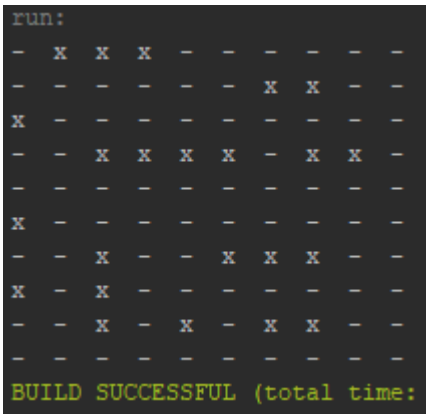
---

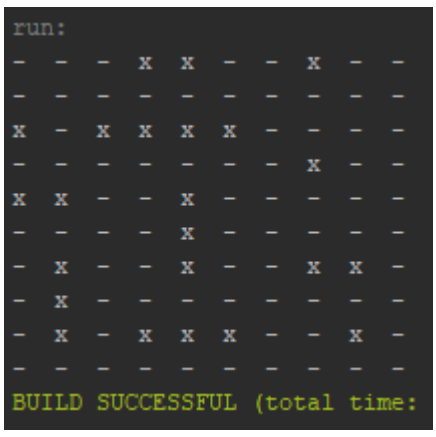
## Testy

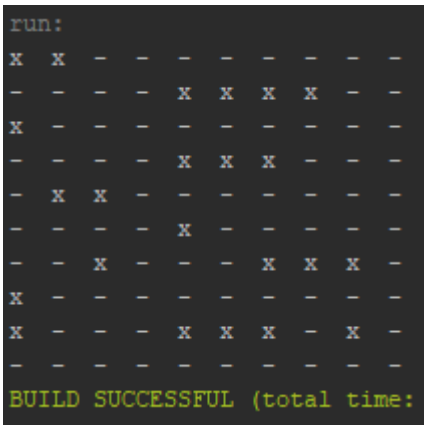
Kilka przykładowych działań programu dla różnych losowych punktów:


run:
- - x - - - - - - -
- - - - x x x - x -
- - x - - - - - - -
x - x - - - - - - -
x - - - x x x - x -
- - - - - - - - x -
x x - - - - - - x -
- - - x x x x - - -
- x - - - - - - x -
- - - - - - - - - -
BUILD SUCCESSFUL (total time:


run:
x - - - - - x - - -
- - x x - - - - - -
- - - - - x x - - -
- - x - - - - - - -
x - x - x x x - x -
x - x - - - - - - -
x - x - - x x - - -
- - - - - - - - - -
- - - x x x - x - -
- - - - - - - - - -
BUILD SUCCESSFUL (total time:


run:
- x x x - - - - - -
- - - - - - x x - -
x - - - - - - - - -
- - x x x x - x x -
- - - - - - - - - -
x - - - - - - - - -
- - x - - x x x - -
x - x - - - - - - -
- - x - x - x x - -
- - - - - - - - - -
BUILD SUCCESSFUL (total time:


run:
- - - x x - - x - -
- - - - - - - - - -
x - x x x x - - - -
- - - - - - - x - -
x x - - x - - - - -
- - - - x - - - - -
- x - - x - - x x -
- x - - - - - - - -
- x - x x x - - x -
- - - - - - - - - -
BUILD SUCCESSFUL (total time:




## Wnioski

Program działa poprawnie natomiast na pewno można nanieść poprawki pod względem długości samego kodu, który z pewnością mógłby być krótszy. Jak i również można by ograniczyć ilość użytych w programie instrukcji warunkowych IF.