UNIWERSYTET KARDYNAŁA STEFANA WYSZYŃSKIEGO W WARSZAWIE

WYDZIAŁ MATEMATYCZNO PRZYRODNICZY. SZKOŁA NAUK ŚCISŁYCH



Projekt Programistyczny Indywidualny

**Gra Snake**

Palina Yeusihneyeva

Warszawa 2022

Spis treści

[Wstęp 2](#_Toc106763681)

[1. Charakterystyka modelu 3](#_Toc106763682)

[1.1 Opis przebiegu gry 4](#_Toc106763683)

[1.2 Opis zastosowanych algorytmów 4](#_Toc106763684)

[1.2.1 Algorytm omijania przeszkód 4](#_Toc106763685)

[1.2.2 Algorytm śledzenia pożywienia 5](#_Toc106763686)

[1.2.3 Algorytm ruchu 7](#_Toc106763687)

[3. Opis programu 9](#_Toc106763688)

[3.1 Środowisko 9](#_Toc106763689)

[3.2 Struktura programu 9](#_Toc106763690)

[3.3 Klasy i struktury danych 9](#_Toc106763691)

[3.4 Schematy blokowe 11](#_Toc106763692)

[3.4.1 Schemat blokowy algorytmu omijania przeszkód 12](#_Toc106763693)

[3.4.2 Schemat blokowy algorytmu śledzenia pożywienia 13](#_Toc106763694)

[3.4.3 Schemat blokowy algorytmu ruchu 14](#_Toc106763695)

[3.5 Opis funkcji i kod źródłowy 15](#_Toc106763696)

[3.6 Interfejs i grafika 22](#_Toc106763697)

[4. Instrukcja obsługi 24](#_Toc106763698)

[4.1 Uruchomienie programu 24](#_Toc106763699)

[4.2 Opis elementów gry 26](#_Toc106763700)

[4.3 Przebieg gry dwuosobowej 28](#_Toc106763701)

[5. Bibliografia 31](#_Toc106763702)

[6. Zawartość archiwum zip 32](#_Toc106763703)

[7. Oświadczenia 32](#_Toc106763704)

# Wstęp

,,Snake” to nazwa popularnej gry komputerowej wydanej po raz pierwszy w 1976 roku pod nazwą Blockade. Została spopularyzowana za pośrednictwem telefonów komórkowych firmy Nokia, w których była dostępna.[[1]](#footnote-1) Dzięki temu zabiegowi tytułowy wąż jest aktualnie jedną z najbardziej rozpoznawalnych gier wideo.

Gra jest stosunkowo prosta i polega na sterowaniu wirtualnym wężem, który poruszając się po planszy zjada kolejne porcje jedzenia i dzięki temu wciąż rośnie. Poza podstawowym elementem zręcznościowym dotyczącym sterowania, skupia się również na logicznym planowaniu trasy, tak aby wąż w nic nie uderzył i zdobył jak największą ilość pokarmu, która przekłada się na większą sumę punktów. Oryginalna gra dotyczyła rozgrywki dwuwymiarowej, natomiast nowsze jej edycje dodają często trzeci wymiar oraz inny tryb śledzenia węża za pośrednictwem np. kamery dynamicznej umieszczonej za jego głową.

Poniższa praca jest dokumentacją indywidualnego projektu gry ,,Snake”. Głównym celem projektu jest zaimplementowanie graficznie gry w środowisku konsolowym wraz z trzema trybami rozgrywki. Program został zaimplementowany w prostym formacie dwuwymiarowym. Projekt w całości został napisany w języku C++ z użyciem następujących bibliotek:

* Iostream
* Windows.h
* Conio.h
* Stdlib.h
* Time.h

Do edycji kodu został wykorzystany edytor Microsoft Visual Studio Community 2022, a do utworzenia diagramów program portal internetowy app.creately.com

# Charakterystyka modelu

Każdy program komputerowy niesie ze sobą różne wyzwania i problemy konieczne do rozwiązania w celu osiągnięcia konkretnej funkcjonalności. Nie inaczej było w przypadku gry Snake.

## Opis przebiegu gry

Gra polega na sterowaniu tytułowym wężem i zjedzeniu jak największej ilości jedzenia. Podczas rozgrywki wąż porusza się jedynie w obrębie ustalonego pola oznaczonego ramką. Każde naprowadzenie głowy węża na jedzenie powoduje automatycznie ,,pożeranie” elementu i wydłużenie węża o stałą wartość (domyślnie grę rozpoczynamy dysponując jedynie głową węża, bez żadnego ogona). Ciało węża przebiega wzdłuż trasy którą nawiguje gracz, dlatego im dłuższy jest wąż tym bardziej precyzyjnie trzeba planować kolejne ruchy. Rozgrywka kończy się gdy wąż uderzy głową w ścianę lub ugryzie samego siebie podczas kolizji. Dodatkowym wyjątkiem (tylko podczas trybu dwuosobowego) jest sytuacja gdy jeden waż ugryzie drugiego.

Każdy zjedzony element zwiększa liczbę punktów gracza w danej próbie. Gracz podczas samodzielnej rozgrywki ma 3 próby. Wygrywa jeśli w ciągu trzech prób osiągnie 10 punktów. Analogicznie przegrywa jeśli nie osiągnie tego wyniku. W trybie gry z drugim graczem lub botem, jeśli ktoś z dwóch graczy nie osiągnie 10 punktów, wygrywa ten gracz, który w ciągu trzech prób zje łącznie więcej elementów.

Sterowanie wężem w trybie jednoosobowym odbywa się za pomocą klawiszy ‘W’, ‘A’, ‘S’, ‘D’, które odpowiadają logicznie kierunkom góra, lewa, dół, prawa. W trybie dwuosobowym, gracz numer jeden gra ponownie z wykorzystaniem klawiszy ‘W’, ‘A’, ‘S’, ‘D’, natomiast gracz drugi kontrolę nad swoim wężem uzyskuje za pośrednictwem przycisków- strzałek góra, lewa, dół, prawa, które działają identycznie jak w pierwszym przykładzie. Komputer steruje wężem samodzielnie.

## Opis zastosowanych algorytmów

### **Algorytm omijania przeszkód**

Algorytm działa w trybie gry z komputerem. Komputer sterując swoim wężem musi unikać zderzenia ze ścianą, własnym ciałem oraz wężem przeciwnika. W związku z tym musi wykrywać przeszkody stojące na jego drodze i wyznaczać nowa trasę bez kolizji.

Algorytm operuje na istniejącej tablicy i zmiennej obierającej kierunek, które są zadeklarowane w atrybutach klasy, dlatego zmienia ich wartości, ustalone na początku gry, nie zwracając niczego.

*Zmienne:*

DIR- kierunek

x- pozycja kolumny

y- pozycja wiersza

mapa- dwuwymiarowa macierz o rozmiarze 40x80, będąca planszą gry, do każdego elementu odnosimy się za pomocą dwóch współrzędnych: y oraz x

*Kroki algorytmu:*

Krok 1. Jeśli element mapy dla współrzędnych x oraz y jest zajęty, przejdź do kroku 2, w przeciwnym wypadku przejdź do kroku 9.

Krok 2. Jeśli DIR równa się dół lub góra przejdź do kroku 3 żeby zmienić kierunek na inny, w przeciwnym wypadku przejdź do kroku 6.

Krok 3. Jeśli element mapy o współrzędnych x-1 oraz y jest wolna, przejdź do kroku 4, w przeciwnym wypadku przejdź do kroku 5.

Krok 4. DIR przyjmuje kierunek lewy, przejdź do kroku 9.

Krok 5. DIR przyjmuje kierunek prawy, przejdź do kroku 9.

Krok 6. Jeśli element mapy o współrzędnych x oraz y-1 jest wolna, przejdź do kroku 7, w przeciwnym wypadku przejdź do kroku 8

Krok 7. DIR przyjmuje kierunek dól, przejdź do kroku 9.

Krok 8. DIR przyjmuje kierunek góra, przejdź do kroku 9.

Krok 9. Uruchom funkcję wykonującą ruch z obranym DIR, zakończ algorytm.

### **Algorytm śledzenia pożywienia**

Algorytm działa w trybie gry z komputerem. Komputer sterując swoim wężem musi szukać jedzenia w obszarze pola gry. W związku z tym musi obierać kierunki trasy tak, aby zdobywać kolejne punkty i powiększać swój ogon.

Algorytm operuje na istniejącej tablicy i zmiennej obierającej kierunek, które są zadeklarowane w atrybutach klasy, dlatego zmienia ich wartości, ustalone na początku gry, nie zwracając niczego.

*Zmienne:*

DIR- kierunek

x- pozycja kolumny

y- pozycja wiersza

i - zmienna sterująca pętlą

mapa- dwuwymiarowa macierz o rozmiarze 40x80, będąca planszą gry, do każdego elementu odnosimy się za pomocą dwóch współrzędnych: y oraz x

*Kroki algorytmu:*

Krok 1. Jeśli DIR przyjmuje wartość dół lub góra, przejdź do kroku 2, w przeciwnym razie przejdź do kroku 7.

Krok 2. Utwórz zmienną i, nadaj jej wartość 0, przejdź do kroku 3.

Krok 3. Jeśli element mapy dla współrzędnych y oraz i zawiera pożywienie, przerwij krok 3 i przejdź do kroku 5. W przeciwnym wypadku przejdź do kroku 4.

Krok 4. Zwiększ wartość zmiennej i o 1 oraz przejdź ponownie do kroku 3.

Krok 5. Jeśli wartość zmiennej i jest mniejsza od rozmiaru szerokości planszy, przejdź do kroku 6.

Krok 6. Jeśli wartość zmiennej i jest mniejsza od wartości x, DIR przyjmuje wartość lewo, w przeciwnym wypadku przyjmuje wartość prawo. Przejdź do kroku 7.

Krok 7. Utwórz zmienną i, nadaj jej wartość 0, przejdź do kroku 8.

Krok 8. Jeśli element mapy dla współrzędnych i oraz x zawiera pożywienie, przerwij krok 8 i przejdź do kroku 10. W przeciwnym wypadku przejdź do kroku 9.

Krok 9. Zwiększ wartość zmiennej i o 1 oraz przejdź ponownie do kroku 8.

Krok 10. Jeśli wartość zmiennej i jest mniejsza od rozmiaru wysokości planszy, przejdź do kroku 11.

Krok 11. Jeśli wartość zmiennej i jest mniejsza od wartości y, DIR przyjmuje wartość dół, w przeciwnym wypadku przyjmuje wartość góra. Przejdź do kroku 12.

Krok 12. Zakończ algorytm i wywołaj algorytm omijania przeszkód.

### **1.2.3 Algorytm ruchu**

Algorytm działa w każdym z trzech trybów gry- dla jednego gracza, dwóch lub rozgrywki z komputerem. Przesuwa on głowę i poruszający się za nią ogon na planszy, sprawdza również czy pole jest puste lub czy znajduje się na nim owoc. Kolizja ze ścianą, własnym ogonem lub drugim graczem zwraca wartość kill (zabity), w przeciwnym razie na końcu algorytmu, zwraca wartość live (żywy).

Algorytm operuje na istniejącej tablicy i zmiennej obierającej kierunek, które są zadeklarowane w atrybutach klasy, dlatego zmienia ich wartość, ustaloną na początku gry, i zwraca jedynie zmienną typu strukturalnego result o wartości kill lub live.

*Zmienne:*

DIR- kierunek

x- pozycja kolumny

y- pozycja wiersza

mapa- dwuwymiarowa macierz o rozmiarze 40x80, będąca planszą gry, do każdego elementu odnosimy się za pomocą dwóch współrzędnych: y oraz x

*Kroki algorytmu:*

Krok 1. Utwórz zmienną o nazwie typu strukturalnego POINT o nazwie old.

Krok 2. Przypisz do pól zmiennej old wartość x oraz y.

Krok 3. Jeśli DIR ma wartość góra to zmniejsz y o 1, idź do kroku 4.

Krok 4. Jeśli DIR ma wartość dół to zwiększ y o 1, idź do kroku 5.

Krok 5. Jeśli DIR ma wartość lewo to zmniejsz x o 1, idź do kroku 6.

Krok 6. Jeśli DIR ma wartość prawo to zwiększ x o 1, idź do kroku 7.

Krok 7. Jeśli element mapy o współrzędnych x i y jest zajęte, zwróć wartość kill i zakończ algorytm, w przeciwnym wypadku idź do kroku 8.

Krok 8. Jeśli element mapy o współrzędnych y i x zawiera pożywienie, idź do kroku 9, w przeciwnym wypadku idź do kroku 11.

Krok 9. Uruchom funkcję, która doda do ogona węża 5 długości- znaków ‘+’ i przesunie go za głową o nowych współrzędnych x oraz y, idź do kroku 10.

Krok 10. Uruchom funkcję, która zwiększy pole eatraenFruites o 1, idź do kroku 12.

Krok 11. Uruchom funkcję, która przesuwa ogon za głową o nowych współrzędnych x oraz y, idź do kroku 12.

Krok 12. Zwróć wartość live (żywy) i zakończ algorytm.

# 3. Opis programu

W tym punkcie zostanie szczegółowo opisana budowa programu oraz jego wszystkie składowe części np. struktury, klasy, metody. Omówione zostaną również schematy blokowe algorytmów opisanych w punkcie 1.2 jak również interfejs graficzny gry.

## 3.1 Środowisko

Program został napisany w całości z wykorzystaniem środowiska Microsoft Visual Studio Community 2022 dla systemu operacyjnego Windows 10, na którym był również testowany. Pliki źródłowe można jednakże uruchomić za pośrednictwem innych środowisk oraz na innych systemach operacyjnych np. dystrybucjach Linuxa.

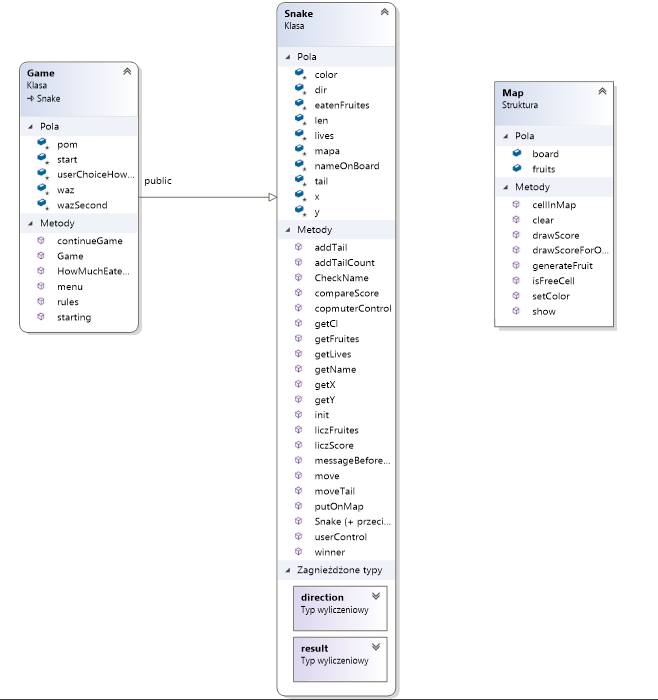
## 3.2 Struktura programu

Program jest podzielony na dwa pliki, jeden o nazwie classesSnake.h zawiera wszystkie deklaracje klas, struktur, pól oraz metod. Posiada również dyrektywy preprocesora #include oraz #define które dołączają pliki bibliotek jak również definiują makra używane w programie np. szerokość i długość planszy.

Drugi plik o nazwie Snake.cpp zawiera kod źródłowy gry oraz ciała wszystkich klas, struktur oraz metod zadeklarowanych w pliku nagłówkowym. To na nim warto się skupić. Program obsługiwany jest przez 33 różne funkcje składowe klas oraz struktur opisane szerzej w punkcie 3.3. ich funkcjonalność jest zróżnicowana, obsługują zarówno wyświetlanie komunikatów dla użytkownika, rysowanie planszy gry, obliczanie punktacji, wyznaczanie zwycięzcy, generowanie pożywienia na mapie, detekcje ruchów i kolizji, algorytm omijania przeszkód, algorytm wyszukiwania pożywienia, wydłużanie ogona węża itp.

## 3.3 Klasy i struktury danych

Projekt skupia się na dwóch klasach- **Snake** i **Game** oraz strukturze **Map**. Poniżej zaprezentowano diagram klas oraz struktur danych wygenerowany przy pomocy narzędzi zestawi Microsoft Visual Studio Community 2022. Zawiera spis wszystkich pól oraz metod jak i zależności klas (dziedziczenia).



Klasa **Snake** to klasa bazowa zawierająca kilka pól dotyczących stanu węża.

* x oraz y to pola całkowite oznaczające pozycję głowy węża
* len zawiera aktualny rozmiar ogona węża
* lives oznacza liczbę żyć/prób gracza
* eatenFruites reprezentuje ilość zjedzonych owoców przez węża
* color zawiera wartość liczbową koloru głowy węża
* direction to typ wyliczeniowy enum, który zawiera cztery możliwe kierunki poruszania się: góra, prawo, lewo, dół
* result to typ wyliczeniowy enum, który zawiera dwa możliwe stany gracza: żywy lub martwy
* tail to tablica typu POINT (struktura zawierająca dwie współrzędne punktu x oraz y) o rozmiarze makra length równym 1000 i oznacza rozmiar ogona węża (który przy jedzeniu wciąż się zwiększa)
* mapa to wskaźnik typu strukturalnego Map reprezentujący mapę- tablicę gry

Klasa **Game** jest klasą pochodną klasy **Snake** i poza dziedziczeniem pól zawiera również własne:

* userChoiceHowToPlay to pole typu całkowitego zawierające decyzję gracza co do wyboru trybu gry (1 to gra jednoosobowa, 2 to gra dla dwóch graczy, 3 to rozgrywka gracza i komputera)
* start to pole typu strukturalnego Map tworzone na początku gry
* pom to pole wskaźnikowe typu strukturalnego Map
* waz oraz wazSecond to pola wskaźnikowe typu Snake oznaczające pierwszego i drugiego gracza

Struktura **Map** reprezentuje planszę gry i zawiera dwa pola:

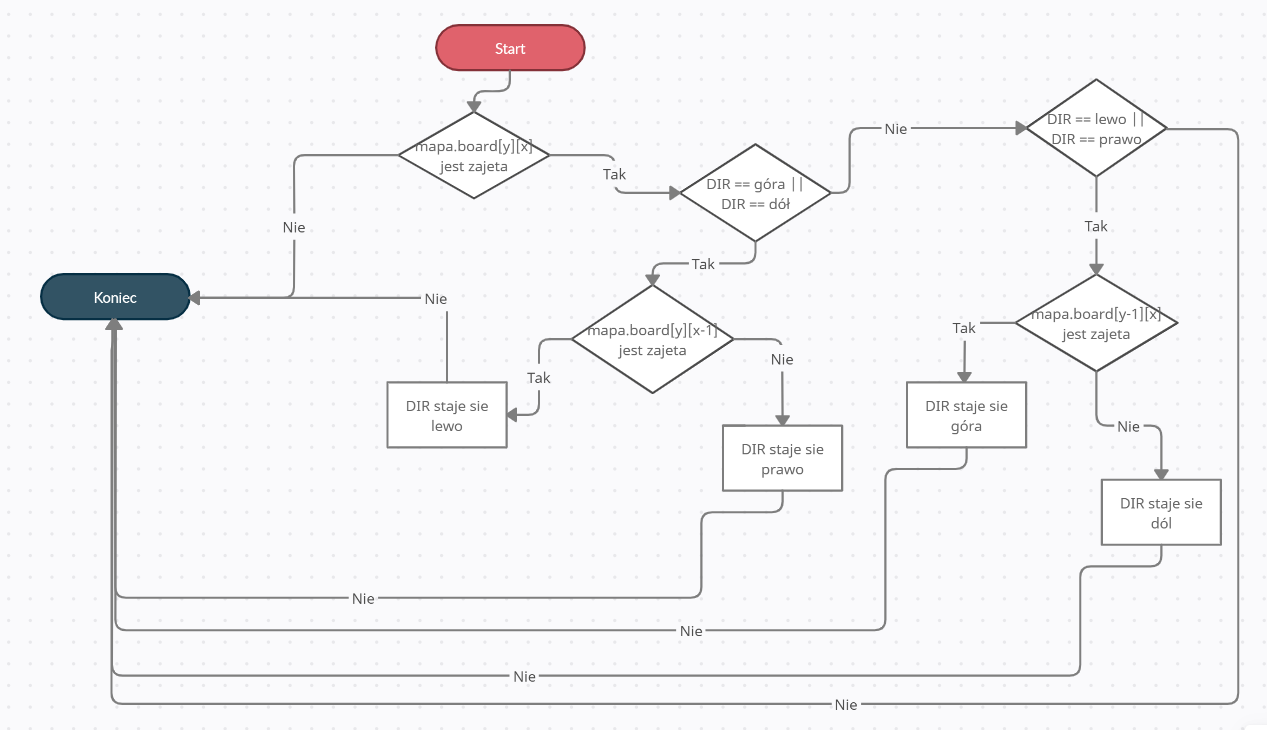
* board to znakowa tablica dwuwymiarowa o rozmiarze 40 na 80 będąca planszą gry w której znajdują się węże oraz pożywienie
* fruits to tablica jednowymiarowa typu strukturalnego POINT zawierająca współrzędne czterech owoców na planszy

## 3.4 Schematy blokowe

Poniżej zaprezentowano schematy blokowe wizualizujące przebieg algorytmów opisanych w punkcie 1.2. Schematy są zaprezentowane pod kątem 90 stopni, dla większej czytelności.

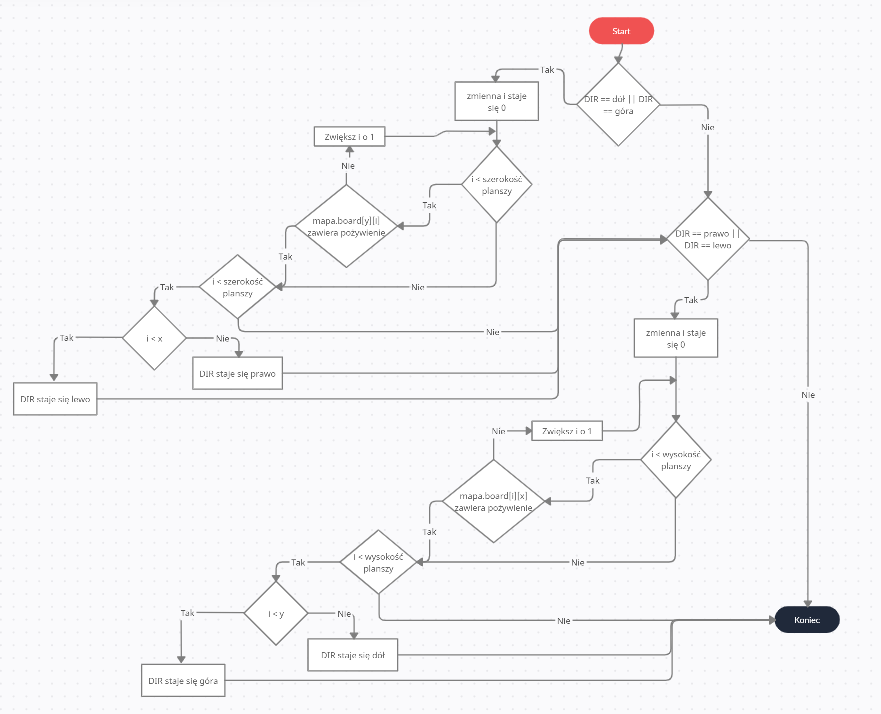
### 

### **3.4.1 Schemat blokowy algorytmu omijania przeszkód**



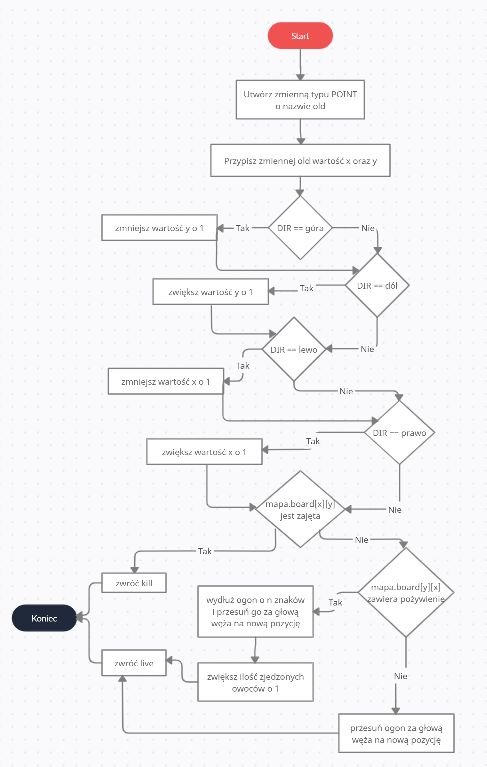
### 

### **3.4.2 Schemat blokowy algorytmu śledzenia pożywienia**



### 

### **3.4.3 Schemat blokowy algorytmu ruchu**



## 

## 3.5 Opis funkcji i kod źródłowy

Kod źródłowy programu składa się z 33 funkcji oraz konstruktorów, geterów i seterów. Poniżej opisano działanie każdej z nich, przy kilku zaprezentowano treść kodu.

Klasa **Snake**:

Snake(); – konstruktor domyślny klasy Snake, tworzy obiekt z podstawowymi wartościami, pustym Nickiem, zeruje ilość punktów i ustawia głowę węża na współrzędne (5,10) z kierunkiem prawo.

Snake(Map\* \_mapa, char name[], int color); -konstruktor argumentowy klasy Snake, przyjmuje wartość strukturalną Map zawierającą dane o planszy, nick oraz kolor głowy węża.

void addTail(int \_x, int \_y); -metoda klasy Snake, nic nie zwraca, dodaje jeden element do ogonu węża, uruchamiając przy tym metodę moveTail(\_x,\_y) i przekazując argumenty dalej.

Obraz zawierający tekst

Opis wygenerowany automatycznie

void moveTail(int \_x, int \_y); - metoda klasy Snake, nic nie zwraca, przesuwa w pętli wartości ogona węża i zmienia współrzędne głowy zgodnie z przyjętymi dwoma argumentami x oraz y po wykonanym ruchu.

Obraz zawierający tekst

Opis wygenerowany automatycznie

void init(int \_x, int \_y, direction \_dir); -metoda klasy Snake, nic nie zwraca, zaczyna grę od nowa aktualizując położenie głowy, usuwając ogon, zerując ilość zjedzonych owoców i wykorzystanych prób jak i obierając przekazany w argumencie kierunek.

void putOnMap(); -metoda klasy Snake, nic nie zwraca, ustawia na mapie głowę i ogon węża, głowa jest reprezentowana przez znak ‘@’ a ogon przez znaki ‘+’. Nie pobiera żadnych danych, ponieważ odczytuje je z pól własnej klasy.

Obraz zawierający tekst

Opis wygenerowany automatycznie

result userControl(char w, char s, char a, char d); - metoda klasy Snake, zwraca typ strukturalny result i przyjmuje 4 wartości typu char. Przechwytuje odczytane dane z klawiatury nadając odpowiednie kierunki wężowi.

Obraz zawierający tekst

Opis wygenerowany automatycznie

void addTailCount(int \_x, int \_y, int count);-metoda klasy Snake, nic nie zwraca, dodaje count elementów do ogona węża o współrzędnych głowy x oraz y, uruchamia funkcję addTail(\_x,\_y).

int liczScore();-metoda klasy Snake, zwraca wartość liczbową całkowitą, zwiększa liczbę wykorzystanych prób węża o 1 .

void liczFruites();-metoda klasy Snake, nic nie zwraca, zwiększa wynik zjedzonych owoców przez węża o 1.

result move();-metoda klasy Snake, zwraca zmienną typu strukturalnego result o wartości live lub kill, opisana szczegółowo w punkcie 1.2.3

Obraz zawierający tekst

Opis wygenerowany automatycznie

result copmuterControl();-metoda klasy Snake, zwraca zmienną typu strukturalnego result o wartości live lub kill, składa się z dwóch algorytmów- omijania przeszkód i śledzenia pożywienia, opisanych szczegółowo w punkcie 1.2.1 oraz 1.2.2

Obraz zawierający tekst

Opis wygenerowany automatycznie

bool winner();-metoda klasy Snake, zwraca wartość true lub false w zależności czy wąż osiągnął 10 punktów (zjedzonych owoców) w próbie.

bool compareScore(Snake& one, Snake& two);-metoda klasy Snake, zwraca wartość true lub false w zależności czy zwyciężył gracz pierwszy(true) czy drugi(false).

int getX(); -metoda klasy Snake, zwraca x.

int getY(); -metoda klasy Snake, zwraca y.

int getCl(); -metoda klasy Snake, zwraca numer koloru.

char\* getName()-metoda klasy Snake, zwraca nick gracza.

int getLives()-metoda klasy Snake, zwraca ilość żyć gracza.

void messageBeforeNewGame(Snake& one);-metoda klasy Snake, nic nie zwraca, przyjmuje referencję do obiektu Snake, informuje o tym który gracz zmarł i doprowadził do kolizji- ze ścianą, własnym ogonem lub ogonem drugiego węża. Metoda działa we wszystkich trzech trybach.

void CheckName(char name[]);-metoda klasy Snake, walidacja wprowadzonego nicku, nie może być dłuższy niż 10 liter.

int getFruites()-metoda klasy Snake, zwraca ilość zjedzonych owoców przez węża.

Klasa **Game**:

Game(); -konstruktor domyślny klasy Game.

void rules();-metoda klasy Game, nic nie zwraca, wypisuje na ekranie zasady gry.

void menu();-metoda klasy Game, nic nie zwraca, wypisuje menu gry (tryb rozgrywek) oraz pozwala na wpisanie numeru wyboru przez gracza, po wyborze uruchamia funkcję starting().

void starting();-metoda klasy Game, nic nie zwraca, jest główną funkcją obsługującą dwa tryby gry w zależności od wyboru gracza, w trybie drugim rozróżnia rozgrywkę z drugim graczem od rozgrywki z komputerem. Zawiera komunikaty o przerwaniu gry po przez kolizje. Czyści ekran konsoli po takim zdarzeniu, tak jak w przypadku planszy.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

void continueGame();-metoda klasy Game, nic nie zwraca, wedle wyboru gracza kontynuuje grę lub usuwa obiekt gracza a wraz z nim jego wynik.

void HowMuchEatenFruites();-metoda klasy Game, nic nie zwraca, wyświetla komunikat który gracz ile zjadł owoców.

Struktura **Map**:

void clear(); -funkcja struktury Map, nic nie zwraca, czyści mapę, ustawia ramy planszy i generuje owoce.

Obraz zawierający tekst

Opis wygenerowany automatycznie

void generateFruit();-funkcja struktury Map, nic nie zwraca, generuje współrzędne dla owoców.

bool isFreeCell(int x, int y);-funkcja struktury Map, zwraca wartość true lub false w zależności czy komórka planszy o współrzędnych x oraz y nie zawiera w sobie znaków ‘@’ oraz ‘+’ (czyli głowy i ciała węża).

bool cellInMap(int x, int y); -funkcja struktury Map, zwraca wartość true lub false w zależności od tego czy komórka o współrzędnych x oraz y mieści się w granicach planszy.

void show(Snake& two); -funkcja struktury Map, nic nie zwraca, wypisuje na ekranie zawartość planszy.

Obraz zawierający tekst

Opis wygenerowany automatycznie

void setColor(int color); -funkcja struktury Map, nic nie zwraca, ustawia kolor tekstu w konsoli wedle przekazanego argumentu color.

void drawScore(Snake& one, Snake& two, int OutTextPosX, int OutTextPosY);-funkcja struktury Map, nic nie zwraca, ustawia nicki i licznik prób na wyświetlanej w konsoli planszy w trybie dla dwóch graczy.

Obraz zawierający tekst

Opis wygenerowany automatycznie

void drawScoreForOnePlayer(Snake& one, int OutTextPosX, int OutTextPosY); -funkcja struktury Map, nic nie zwraca, ustawia nick i licznik prób na wyświetlanej w konsoli planszy w trybie dla jednego gracza.

Główna funkcja **main()**

Tworzy obiekt typu Game i wywołuje na jego rzecz metodę menu().

## 3.6 Interfejs i grafika

Interfejs gry jest uruchamiany w konsoli tak jak i układ graficzny gry. Komunikacja z użytkownikiem odbywa się po przez wypisywanie na ekranie konsoli komunikatów i wpisywanie przez gracza konkretnych decyzji zatwierdzanych Enterem np. wpisywanie liczb od 1 do 3 przy wyborze trybu gry czy wpisanie nicku gracza. Podczas gry użytkownik steruje wężem za pomocą klawiszy ‘W’, ‘A’, ‘S’, ‘D’, które odpowiadają logicznie kierunkom góra, lewa, dół, prawa.

Grafika gry złożona jest z tekstowej reprezentacji planszy oraz innych elementów.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Napis „Palina: 0 | Ewa: 0” u góry ekranu oznacza imię pierwszego gracza (podanego przez użytkownika) oraz liczbę jego wykorzystanych prób. Analogicznie Ewa: 0 oznacza zapisane imię drugiego gracza oraz jego liczbę wykorzystanych prób. Przy starcie gry (w tym przypadku w trybie dwuosobowym) każdy gracz ma zerowy licznik prób, przy popełnieniu błędu np. kolizji ze ścianą, gracz traci próbę i jego licznik zwiększa się o 1. Każdy gracz ma maksymalnie 3 próby, po ich przekroczeniu gra się kończy i zwycięża ta osoba, która ma najwięcej punktów.

Znak ‘–‘ oraz ‘|’ oznaczają ramy pola gry.

Znak ‘@’ oznacza głowę węża- zarówno pierwszego jak i drugiego, różnią się jedynie kolorem.

Znak ‘\*’ oznacza jedzenie (owoce) umieszczone na planszy.

Znak ‘+’ oznacza ogon węża.

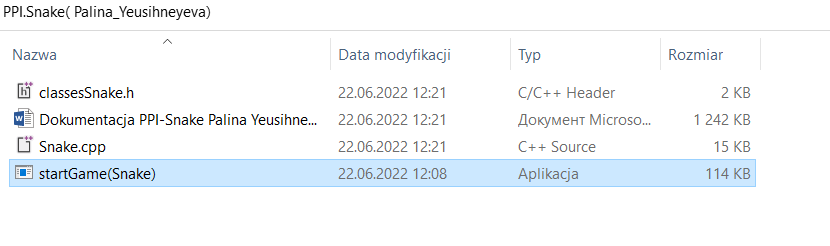
# 

# 4. Instrukcja obsługi

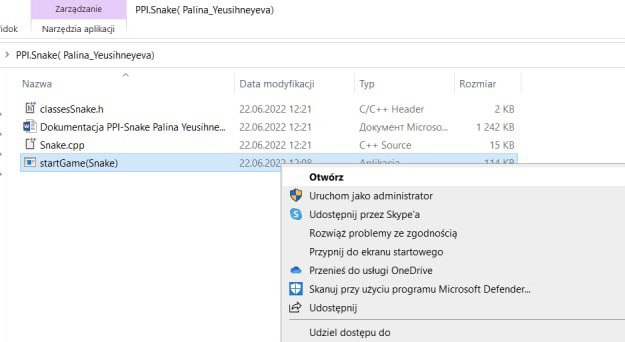
W celu lepszego zrozumienia gry oraz wszystkich funkcjonalności, które oferuje, zostaną tu omówione aspekty uruchomienia programu oraz przebiegi gry.

## 4.1 Uruchomienie programu

Dla uruchomenia gry na początku należy znalezć plik “startGame(Snake).exe” w rozpakowanym folderze “PPI.Snake(Palina\_Yeusihneyeva).zip”.



Następnie należy kliknąć prawym przyciskiem myszy na wybrany plik I wybrać opcję “otwórz”.



Po uruchomieniu programu widzimy nowo otwarte okno konsoli, którą należy rozszerzyć na cały ekran, i następujący komunikat:



Oznacza to że gra uruchomiła się poprawnie i możemy już przetestować program. Dla początkujących użytkowników, zalecane jest wybranie punktu pierwszego w celu przeczytania opisu i zasad gry:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Bezpośrednio pod rozpisanymi zasadami można wybrać tryb gry. 1 To gra jednoosobowa, 2 to gra dla dwóch graczy, 3 to rozgrywka gracza z komputerem. Możliwe jest również wybranie od razu punktu drugiego, bez czytania zasad:

Obraz zawierający tekst

Opis wygenerowany automatycznie

## 

## 4.2 Opis elementów gry

Gra składa się z trzech trybów rozgrywki: jednoosobowej, dwuosobowej i gry gracza z komputerem.

Rozgrywka jednoosobowa polega na sterowaniu jednym wężem i próbie zdobycia 10 punktów podczas 3 prób. Każda próba kończy się gdy gracz najedzie głową węża na ścianę lub na jego ogon.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rozgrywka dwuosobowa polega na sterowaniu dwoma wężami przez dwóch graczy. Wygrywa ten gracz, który po 3 próbach ma większą ilość punktów. Próba kończy się gdy któryś z węży uderzy głową w ścianę, ugryzie własny ogon lub ogon przeciwnika.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Rozgrywka gracza z komputerem jest identyczna jak w przypadku gry dwuosobowej, z tą różnicą, że drugim wężem operuje komputer.

Obraz zawierający tekst, sprzęt elektroniczny

Opis wygenerowany automatycznie

## 

## 4.3 Przebieg gry dwuosobowej

Po uruchomieniu programu i wybraniu opcji gry dwuosobowej widzimy następujący komunikat:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Uzupełniamy imiona pierwszego i drugiego gracza. Następnie wyświetla się plansza wraz z dwoma wężami. Gracz pierwszy oznaczony jest białym znakiem @ natomiast drugi zielonym znakiem @. Obie głowy węży poruszają się domyślnie w prawo i lewo, w tym momencie kontrolę nad nimi przejmują gracze.

Obraz zawierający tekst, zrzut ekranu, nocne niebo

Opis wygenerowany automatycznie

Sterowanie wężem w trybie dwuosobowym odbywa się za pomocą klawiszy, gracz numer jeden gra z wykorzystaniem przycisków ‘W’, ‘A’, ‘S’, ‘D’, które odpowiadają kierunkom góra, lewa, dół, prawa. Natomiast gracz drugi kontrolę nad swoim wężem uzyskuje za pośrednictwem przycisków- strzałek góra, lewa, dół, prawa, które działają identycznie jak w pierwszym przykładzie.

Możliwe są kolizje, które przerywają daną próbę i informują o błędzie graczy:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Wygrywa gracz który pierwszy osiągnie 10 punktów.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Zgodnie z poleceniem grę można zakończyć lub kontynuować.

# 5. Bibliografia

1. https://pl.wikipedia.org/wiki/W%C4%85%C5%BC\_(gra\_komputerowa)

2. https://docs.microsoft.com/en-us/windows/console/setconsolecursorposition?fbclid=IwAR2I1fTzfS\_D62Ya4A-5A3oIY9fKq\_VowBScIMYlri6OpvMqOzW8lnr-ZDU

3. https://stackoverflow.com/questions/4053837/colorizing-text-in-the-console-with-c?fbclid=IwAR0wlRl0htJ0ncG8iU-ltYFtfXJha8vDHpTXb1n5ErS3\_wiHHFBfB\_YxFzU

4. https://docs.microsoft.com/en-us/windows/win32/api/windef/ns-windef-point

5. https://stackoverflow.com/questions/24708700/c-detect-when-user-presses-arrow-key

# 6. Zawartość archiwum zip

Archiwum zip składa się z trzech plików:

* classesSnake.h
* Snake.cpp
* Dokumentacja PPISnake Palina Yeusihneyeva.docx
* startGame(Snake).exe

Dwa pierwsze pliki zawierają kod źródłowy programu, który po skompilowaniu (np. przy wykorzystaniu Microsoft Visual Studio 2022) zostanie uruchomiony jako plik wykonywalny o rozszerzeniu .exe (dla systemu Windows). Program zostanie uruchomiony w konsoli systemowej.

Plik **classesSnake.h** zawiera deklaracje wszystkich struktur, klas oraz ich metod zaimplementowanych w projekcie. Posiada również dyrektywy preprocesora- Include, dodające pliki bibliotek.

Plik **Snake.cpp** zawiera kod źródłowy gry oraz ciała wszystkich klas, struktur oraz metod. Zawiera również funkcje główną main(), która uruchamia grę.

Plik **Dokumentacja PPISnake Palina Yeusihneyeva.docx** jest dokładnie tym dokumentem opisującym szczegółowo cały projekt.

Plik **startGame(Snake).exe** jest aplikacją gry, za pomocą ktorego można uruchomić grę.

# 7. Oświadczenia

Świadoma odpowiedzialności prawnej oświadczam, że niniejszy projekt został napisany przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego lub zaliczenia przedmiotu z żadnej uczelni ani szkoły.

Poniżej umieszczam skan pisemnego oświadczenia będącego Załącznikiem nr 1 do Decyzji Nr 10/2020 z dnia 2 czerwca 2020 r. Prorektora ds. Studenckich i Kształcenia.

1. https://pl.wikipedia.org/wiki/W%C4%85%C5%BC\_(gra\_komputerowa) dostęp 13.06.2022r. [↑](#footnote-ref-1)