

# **PacManClone**

AUTHOR

Версия

Ср Май 18 2022



# Съдържание

Table of contents



# Именни пространства Указател

## Именни пространства Списък

пълен списък с именни пространства с кратко описание:

<b>РасМан</b> .....	5
---------------------	---

# Класове Указател

## Класове Списък

Класове, структури, обединения и интерфейси с кратко описание:

<b>PacMan::Game</b> .....	6
<b>PacMan::Ghost</b> .....	7
<b>PacMan::Map&lt; SizeX, SizeY &gt;</b> .....	9
<b>PacMan::Player</b> .....	11
<b>PacMan::Point</b> .....	13

# Файлове Списък

## Файлове Списък

Пълен списък с файлове с кратко описание:

<b>PacManClone/Ghost.h</b> .....	15
<b>PacManClone/main.cpp</b> .....	21
<b>PacManClone/Map.cpp</b> .....	22
<b>PacManClone/Map.h</b> .....	23
<b>PacManClone/Math.cpp</b> .....	25
<b>PacManClone/Math.h</b> .....	26
<b>PacManClone/PacMan.cpp</b> .....	28
<b>PacManClone/PacMan.h</b> .....	29
<b>PacManClone/Player.cpp</b> .....	31
<b>PacManClone/Player.h</b> .....	32
<b>PacManClone/Point.h</b> .....	34
<b>PacManClone/WindowsFunctions.cpp</b> .....	36
<b>PacManClone/WindowsFunctions.h</b> .....	37

# Именни пространства Документация

## РасМан Именно пространството Справка

### Класове

class **Game**  
class **Ghost**  
class **Map**  
class **Player**  
class **Point**



# Класове Документация

## РacMan::Game Клас Препратка

```
#include <PacMan.h>
```

### Общодостъпни членове функции

```
Game ()=default  
void Init ()  
void Update ()
```

---

### Конструктор & Деструктор Документация

```
РacMan::Game::Game () [default]
```

---

### Членове Функции(методи) Документация

```
void РacMan::Game::Init ()
```

```
void РacMan::Game::Update ()
```

---

### Документация за клас генериран от следните файлове:

```
PacManClone/РacMan.h  
PacManClone/РacMan.cpp
```

## РacMan::Ghost Клас Препратка

```
#include <Ghost.h>
```

### Общодостъпни членове функции

```
Ghost ()=default  
void Move (Point dir)  
float GetTargetDistance (uint32_t direction)  
bool IsCollidingPacman (Player &player)  
void UpdateTarget (uint32_t pacmanDirection, Point ghostPosition, Point pacmanPosition, Map< 21,  
21 > &map)  
void Update (Map< 21, 21 > &map, Player &player, Ghost &ghost)
```

### Общодостъпни атрибути

```
uint32_t id {}  
uint32_t frightenedSpeedTimer {}  
bool frightenedMode {}  
uint32_t movementMode {0}  
Point position {}  
Point home {10,10}  
uint32_t direction {}  
Point target {}
```

---

## Конструктор & Деструктор Документация

**РacMan::Ghost::Ghost () [default]**

---

## Членове Функции(методи) Документация

**float PacMan::Ghost::GetTargetDistance (uint32\_t direction) [inline]**

Функция която връща разстоянието до целта

**bool PacMan::Ghost::IsCollidingPacman (Player & player) [inline]**

Функция която проверява дали има колизия м-у духчето и пакман

**void PacMan::Ghost::Move (Point dir) [inline]**

Функция за движение на духчето в някаква посока

**void PacMan::Ghost::Update (Map< 21, 21 > & map, Player & player, Ghost & ghost) [inline]**

Функция за опресняване на духчето

**void PacMan::Ghost::UpdateTarget (uint32\_t pacmanDirection, Point ghostPosition, Point pacmanPosition, Map< 21, 21 > & map) [inline]**

Функция която опреснява целта

---

## Член данни Документация

**uint32\_t PacMan::Ghost::direction {}**

Посока на духчето

**bool PacMan::Ghost::frightenedMode {}**

Променлива която служи за проверка дали е уплашено

**uint32\_t PacMan::Ghost::frightenedSpeedTimer {}**

Таймер за колко време се страхува

**Point PacMan::Ghost::home {10,10}**

Позиция на къщата на духчето

**uint32\_t PacMan::Ghost::id {}**

Идентификатор на духчето

**uint32\_t PacMan::Ghost::movementMode {0}**

Режим на движение

**Point PacMan::Ghost::position {}**

Позиция на духчето

**Point PacMan::Ghost::target {}**

Цел на духчето

---

Документация за клас генериран от следният файл:

PacManClone/Ghost.h

## Пacman::Map< SizeX, SizeY > Клас Шаблон Препратка

```
#include <Map.h>
```

### Общодостъпни членове функции

```
Map ()=default  
void Set (size_t x, size_t y, int8_t value)  
void Set (Point pos, int8_t value)  
int8_t Get (size_t x, size_t y) const  
int8_t Get (Point pos) const  
bool IsValid (Point pos)  
auto & data ()  
auto GetSizeX () const  
auto GetSizeY () const  
const void Draw (Point playerPosition, Point ghostPosition[4], bool frightened[4]) const  
void Fill (int8_t symbol)  
void Fill (std::string data[SizeY])  
bool isColliding (Point p)
```

---

### Конструктор & Деструктор Документация

```
template<uint32_t SizeX, uint32_t SizeY> Pacman::Map< SizeX, SizeY >::Map ()  
[default]
```

Конструктор по подразбиране

---

### Членове Функции(методи) Документация

```
template<uint32_t SizeX, uint32_t SizeY> auto & Pacman::Map< SizeX, SizeY >::data ()  
[inline]
```

Функция която връща низа с информация на картата

```
template<uint32_t SizeX, uint32_t SizeY> const void Pacman::Map< SizeX, SizeY  
>::Draw (Point playerPosition, Point ghostPosition[4], bool frightened[4]) const
```

Функция която рисува

```
template<uint32_t SizeX, uint32_t SizeY> void Pacman::Map< SizeX, SizeY >::Fill (int8_t  
symbol)
```

Функция която запълва картата с някакъв символ

```
template<uint32_t SizeX, uint32_t SizeY> void Pacman::Map< SizeX, SizeY >::Fill  
(std::string data[SizeY])
```

```
template<uint32_t SizeX, uint32_t SizeY> int8_t Pacman::Map< SizeX, SizeY >::Get  
(Point pos) const[inline]
```

Функция която връща стойност на клетка с позиция pos

```
template<uint32_t SizeX, uint32_t SizeY> int8_t Pacman::Map< SizeX, SizeY >::Get  
(size_t x, size_t y) const[inline]
```

Функция която връща стойност на клетка с позиция x,y

```
template<uint32_t SizeX, uint32_t SizeY> auto PacMan::Map< SizeX, SizeY >::GetSizeX  
( ) const[inline]
```

Функция която връща широчина на картата

```
template<uint32_t SizeX, uint32_t SizeY> auto PacMan::Map< SizeX, SizeY >::GetSizeY  
( ) const[inline]
```

Функция която връща височина на картата

```
template<uint32_t SizeX, uint32_t SizeY> bool PacMan::Map< SizeX, SizeY  
>::isColliding (Point p)[inline]
```

Функция която проверява дали позицията се намира на мястото на стена

```
template<uint32_t SizeX, uint32_t SizeY> bool PacMan::Map< SizeX, SizeY >::IsValid  
(Point pos)[inline]
```

Функция която проверява дали точката е в картата

```
template<uint32_t SizeX, uint32_t SizeY> void PacMan::Map< SizeX, SizeY >::Set (Point  
pos, int8_t value)[inline]
```

Функция която задава стойност на клетка с позиция pos

```
template<uint32_t SizeX, uint32_t SizeY> void PacMan::Map< SizeX, SizeY >::Set  
(size_t x, size_t y, int8_t value)[inline]
```

Функция която задава стойност на клетка с позиция x,y

---

**Документация за клас генериран от следните файлове:**

PacManClone/Map.h

PacManClone/Map.cpp

## РacMan::Player Клас Препратка

```
#include <Player.h>
```

### Общодостъпни членове функции

```
Player ()=default  
Player (const Point startPosition)  
void Move (Point dir)  
void Update (Map< 21, 21 > &map, Point ghostPositions[4])
```

### Общодостъпни атрибути

```
bool isDead {}  
Point position {}  
uint32_t direction {}  
uint32_t score {}  
uint32_t scoreMultiplier {}  
uint32_t multiplierTicks {}  
bool isPowered {}
```

---

### Подробно описание

Клас Играч

---

### Конструктор & Деструктор Документация

**РacMan::Player::Player () [default]**

Конструктор по подразбиране

**РacMan::Player::Player (const Point *startPosition*) [inline]**

Конструктор при който се задава началната позиция на играча

---

### Членове Функции(методи) Документация

**void РacMan::Player::Move (Point *dir*) [inline]**

Функция която мърда играча в дадена посока

**void РacMan::Player::Update (Map< 21, 21 > & *map*, Point *ghostPositions*[4])**

Функция която опреснява играча и картата

---

### Член данни Документация

**uint32\_t РacMan::Player::direction {}**

Посока в която се движи играча

**bool РacMan::Player::isDead {}**

Флаг който служи за проверка дали играчът е умрял

**bool PacMan::Player::isPowered {}**

Флаг за проверка дали може да изяде духче

**uint32\_t PacMan::Player::multiplierTicks {}**

Изминали тикове

**Point PacMan::Player::position {}**

Позиция на играча

**uint32\_t PacMan::Player::score {}**

Точки които е събрал играча

**uint32\_t PacMan::Player::scoreMultiplier {}**

умножител на точки

---

**Документация за клас генериран от следните файлове:**

PacManClone/**Player.h**

PacManClone/**Player.cpp**

## РасМан::Point Клас Препратка

```
#include <Point.h>
```

### Общодостъпни членове функции

```
Point ()=default  
Point (float x, float y)  
Point (const Point &)=default  
Point & operator= (const Point &other)=default  
bool operator== (const Point &rhs)  
Point & operator+= (const Point &rhs)  
Point & operator-= (const Point &rhs)  
bool operator!= (const Point &rhs)  
Point operator+ (const Point &rhs)  
Point & operator* (float rhs)  
Point GetTrunc ()
```

### Общодостъпни атрибути

```
float x {}  
float y {}
```

---

## Конструктор & Деструктор Документация

**РасМан::Point::Point () [default]**

Конструктор по подразбиране

**РасМан::Point::Point (float x, float y) [inline]**

Конструктор който взима x,y като параметри

**РасМан::Point::Point (const Point & ) [default]**

Конструктор който копира данните от друга точка

---



#### Членове Функции(методи) Документация

**Point PacMan::Point::GetTrunc () [inline]**

**bool PacMan::Point::operator!= (const Point & *rhs*) [inline]**

**Point & PacMan::Point::operator\* (float *rhs*) [inline]**

**Point PacMan::Point::operator+ (const Point & *rhs*) [inline]**

**Point & PacMan::Point::operator+= (const Point & *rhs*) [inline]**

**Point & PacMan::Point::operator-= (const Point & *rhs*) [inline]**

**Point & PacMan::Point::operator= (const Point & *other*) [default]**

**bool PacMan::Point::operator== (const Point & *rhs*) [inline]**

---

#### Член данни Документация

**float PacMan::Point::x {}**

**float PacMan::Point::y {}**

---

**Документация за клас генериран от следният файл:**

**PacManClone/Point.h**

# Файлове Документация

## РасManClone/Ghost.h Файл Справка

```
#include "Point.h"  
#include <iostream>  
#include <stdlib.h>  
#include "Map.h"  
#include "Math.h"  
#include "Player.h"
```

### Класове

```
class РасMan::Ghost
```

### Именни пространства

```
namespace РасMan
```

## PacManClone/Ghost.h

Вижте документацията за този файл.1 #pragma once

```
2 #include "Point.h"
3 #include <iostream>
4 #include <stdlib.h>
5 #include "Map.h"
6 #include "Math.h"
7 #include "Player.h"
8
9 namespace PacMan {
10     class Ghost {
11     public:
12         Ghost() = default;
13
14         void Move(Point dir) { position += dir; }
15
16         float GetTargetDistance(uint32_t direction) {
17             float x = position.x;
18             float y = position.y;
19
20             switch (direction)
21             {
22             case 0:
23             {
24                 x += 1;
25
26                 break;
27             }
28             case 1:
29             {
30                 y -= 1;
31
32                 break;
33             }
34             case 2:
35             {
36                 x -= 1;
37
38                 break;
39             }
40             case 3:
41             {
42                 y += 1;
43
44                 break;
45             }
46             }
47             return GetDistance({ x,y }, target);
48         }
49
50         bool IsCollidingPacman(Player& player) {
51             return player.position.GetTrunc() == position;
52         }
53
54         void UpdateTarget(uint32_t pacmanDirection, Point ghostPosition, Point
pacmanPosition, Map<21,21>& map) {
55             if (movementMode == 0) // scatter mode
56             {
57                 //отива в края на картата
58                 switch (id)
59                 {
60                 case 0:
61                 {
62                     target = {(map.GetSizeX() - 1.0f), 0.0f };
63
64                     break;
65                 }
66                 case 1:
67                 {
68                     target = { 0.0f, 0.0f };
69
70                     break;
71                 }
72                 case 2:
73                 {
74                     target = { (map.GetSizeX() - 1.0f), (map.GetSizeY()
- 1.0f) };
75                 }
76             }
77         }
78     }
79 }
```

```

84
85         break;
86     }
87     case 3:
88     {
89         target = { 0.0f, (map.GetSizeY() - 1.0f) };
90     }
91     }
92 }
93 else //chase mode
94 {
95     switch (id)
96     {
97     case 0: //червеният дух нацелва пакман
98     {
99         target = pacmanPosition;
100
101         break;
102     }
103     case 1: //розовият дух нацелва 4 клетки пред пакман
104     {
105         target = pacmanPosition;
106
107         switch (pacmanDirection)
108         {
109         case 0:
110         {
111             target += {2, 0};
112
113             break;
114         }
115         case 1:
116         {
117             target -= {0, 2};
118
119             break;
120         }
121         case 2:
122         {
123             target -= {2, 0};
124             break;
125         }
126         case 3:
127         {
128             target += {0, 2};
129
130         }
131         }
132
133         break;
134     }
135     case 2: // синият нацелва 2 клетки пред пакман
136     {
137         target = pacmanPosition;
138
139         switch (pacmanDirection)
140         {
141         case 0:
142         {
143             target += {1, 0};
144
145             break;
146         }
147         case 1:
148         {
149             target -= {0, 1};
150
151             break;
152         }
153         case 2:
154         {
155             target -= {1, 0};
156
157             break;
158         }
159         case 3:
160         {

```

```

161             target += {0, 1};
162         }
163     }
164
165     target += {target.x - ghostPosition.x,
166               target.y - ghostPosition.y};
167
168     break;
169 }
170 case 3: // оранжевият дух преследва пакман и след това
отива в scatter режим
171 {
172
173     if ( GetDistance(pacmanPosition,position) >= 4)
174     {
175         target = pacmanPosition;
176     }
177     else
178     {
179         target = { 0.0f, (map.GetSizeY() - 1.0f) };
180     }
181     }
182 }
183 }
184
185 void Update(Map<21, 21>& map, Player& player,Ghost& ghost) {
186     bool move = 0;
187
188     unsigned char available_ways = 0;
189     float speed = 1;
190
191     bool walls[4]{};
192
193     UpdateTarget(player.direction,ghost.position,
194 player.position,map);
195
196     walls[0] = map.Get(position + Point{ speed,0 }) == '#';
197     walls[1] = map.Get(position + Point{ 0,-speed }) == '#';
198     walls[2] = map.Get(position + Point{ -speed,0 }) == '#';
199     walls[3] = map.Get(position + Point{ 0,speed }) == '#';
200
201     if (frightenedMode != 1)
202     {
203         unsigned char optimal_direction = 4;
204
205         // духчето може да мърда
206         move = 1;
207
208         for (unsigned char a = 0; a < 4; a++)
209         {
210             //духчетата не могат да се обърнат, стига да не е нужно
211             if (a == (2 + direction) % 4)
212             {
213                 continue;
214             }
215             else if (walls[a] == false)
216             {
217                 if (optimal_direction == 4)
218                 {
219                     optimal_direction = a;
220                 }
221                 available_ways++;
222                 if (GetTargetDistance(a) <
223 GetTargetDistance(optimal_direction))
224                 {
225                     // оптималната посока за да стигне целта
226                     optimal_direction = a;
227                 }
228             }
229         }
230
231         if (available_ways > 1)
232         {
233             // когато стигне кръстопът избира най добрата посока
234             direction = optimal_direction;
235         }
236     }
237 }

```

```

238     }
239     else
240     {
241         if (optimal_direction == 4)
242         {
243             // обръща се духчето на обратно защото трябва
244             direction = (2 + direction) % 4;
245         }
246         else
247         {
248             direction = optimal_direction;
249         }
250     }
251 }
252 else
253 {
254     unsigned char random_direction = rand() % 4;
255
256     if (frightenedSpeedTimer == 0)
257     {
258         move = 1;
259
260         frightenedSpeedTimer = 30;
261
262         for (unsigned char a = 0; a < 4; a++)
263         {
264             // не може да се обръщат дори и да се страхуват
265             if (a == (2 + direction) % 4)
266             {
267                 continue;
268             }
269             else if (walls[a] == 0)
270             {
271                 available_ways++;
272             }
273         }
274
275         if (available_ways > 0)
276         {
277             while (walls[random_direction] == 1 ||
278 random_direction == (2 + direction) % 4)
279             {
280                 // вземаме произволна посока докато не може да
281                 random_direction = rand() % 4;
282             }
283
284             direction = random_direction;
285         }
286         else
287         {
288             // ако няма друг път се обръща наобратно
289             direction = (2 + direction) % 4;
290         }
291     }
292     else
293     {
294         frightenedSpeedTimer--;
295     }
296 }
297 // ако може да се мръдне го местим
298 if (move)
299 {
300     switch (direction)
301     {
302     case 0:
303     {
304         position.x += speed;
305
306         break;
307     }
308     case 1:
309     {
310         position.y -= speed;
311
312         break;

```

```

313         }
314         case 2:
315         {
316             position.x -= speed;
317
318             break;
319         }
320         case 3:
321         {
322             position.y += speed;
323         }
324     }
325
326     // когато мине през портала го пращаме от другата страна
327     if (position.x <= -1)
328     {
329         position.x = map.GetSizeX() - speed;
330     }
331     else if (position.x >= map.GetSizeX())
332     {
333         position.x = speed;
334     }
335 }
336
337 if (position.GetTrunc() == player.position.GetTrunc())
338 {
339     if (frightenedMode == 0) // когато не е уплашен и докосне
340 пакман, играта приключва
341     {
342         player.isDead = 1;
343     }
344     else //иначе бяга към клетката
345     {
346         frightenedMode = 2;
347
348         target = home;
349     }
350 }
351
352 uint32_t id{};
353 uint32_t frightenedSpeedTimer{};
354 bool frightenedMode{};
355 uint32_t movementMode{0};
356 Point position{};
357 Point home{10,10};
358 uint32_t direction{};
359 Point target{};
360 };
361 }

```

## РacManClone/main.cpp Файл Справка

```
#include <iostream>
#include <Windows.h>
#include "PacMan.h"
#include "WindowsFunctions.h"
```

### Функции

int **main** ()

---

### Функции Документация

int **main** ()



## **РасManClone/Map.cpp Файл Справка**

```
#include "Map.h"  
#include <iostream>  
#include <Windows.h>  
#include "WindowsFunctions.h"
```

### **Именни пространства**

```
namespace РасMan
```

## **РacManClone/Map.h Файл Справка**

```
#include <cstdint>
#include <string>
#include "Point.h"
```

### **Класове**

```
class РacMan::Map< SizeX, SizeY >
```

### **Именни пространства**

```
namespace РacMan
```

## PacManClone/Map.h

Вижте документацията за този файл.1 #pragma once

```
2 #include <cstdint>
3 #include <string>
4 #include "Point.h"
5
6 namespace PacMan {
7
8     template <uint32_t SizeX, uint32_t SizeY>
9     class Map {
10     public:
11         Map() = default;
12
13         void Set(size_t x, size_t y, int8_t value) { m_Data[x][y] = value; }
14         void Set(Point pos, int8_t value) { m_Data[(int)pos.x][(int)pos.y] =
value; }
15
16         int8_t Get(size_t x, size_t y) const { return m_Data[x][y]; }
17         int8_t Get(Point pos) const {
18             if (pos.y < height && pos.x < width && pos.x >= 0 && pos.y >= 0) {
19                 return m_Data[(size_t)pos.x][(size_t)pos.y];
20             }
21             return INT8_MIN;
22         }
23
24         bool IsValid(Point pos) { return (pos.y < height && pos.x < width &&
pos.x >= 0 && pos.y >= 0); }
25         auto& data() { return m_Data; }
26
27         auto GetSizeX() const { return SizeX; };
28         auto GetSizeY() const { return SizeY; };
29         const void Draw(Point playerPosition, Point ghostPosition[4], bool
frightened[4]) const;
30         void Fill(int8_t symbol);
31
32         /*
33          * Функция която запълва картата с някакъв низ от редове
34          */
35         void Fill(std::string data[SizeY]);
36
37         bool isColliding(Point p) {
38             char c = m_Data[(int)p.GetTrunc().x][(int)p.GetTrunc().y];
39             return c == '#';
40         }
41     private:
42         char m_Data[SizeX][SizeY]{};
43         uint32_t width = SizeX;
44         uint32_t height = SizeY;
45     };
46 }
```

## РacManClone/Math.cpp Файл Справка

```
#include "Math.h"
```

### Функции

```
float GetDistance (РacMan::Point p1, РacMan::Point p2)
```

---

### Функции Документация

**float GetDistance (РacMan::Point *p1*, РacMan::Point *p2*)**

Функция за изчисляване на дистанция използвайки теоремата на Питагор

## РacManClone/Math.h Файл Справка

```
#include "Point.h"
```

### Функции

```
float GetDistance (РacMan::Point p1, РacMan::Point p2)
```

---

### Функции Документация

```
float GetDistance (РacMan::Point p1, РacMan::Point p2)
```

Функция за изчисляване на дистанция използвайки теоремата на Питагор

## PacManClone/Math.h

Вижте документацията за този файл.1 `#pragma once`  
2 `#include "Point.h"`  
6 `float GetDistance(PacMan::Point p1, PacMan::Point p2);`

## **РасManClone/РасMan.cpp Файл Справка**

```
#include "PacMan.h"  
#include "Map.cpp"  
#include <Windows.h>  
#include "WindowsFunctions.h"
```

## **РасManClone/РасMan.h Файл Справка**

```
#include "Player.h"  
#include "Map.h"  
#include "Ghost.h"
```

### **Класове**

```
class РасMan::Game
```

### **Именни пространства**

```
namespace РасMan
```



## PacManClone/PacMan.h

Вижте документацията за този файл.1 #pragma once

```
2 #include "Player.h"
3 #include "Map.h"
4 #include "Ghost.h"
5
6 namespace PacMan {
7     class Game {
8     public:
9         /*
10          * Конструктор по подразбиране
11          */
12         Game() = default;
13         /*
14          * Функция за инициализация на играта
15          */
16         void Init();
17         /*
18          * Функция която опреснява играта
19          */
20         void Update();
21     private:
22         /*
23          * Предишна позиция на играча
24          */
25         Point m_OldPlayerPosition{};
26         /*
27          * Играч
28          */
29         Player m_Player{};
30         /*
31          * Низ от духчета които се движат на картата
32          */
33         Ghost m_Enemy[4]{};
34         /*
35          * Карта с размери [21,21]
36          */
37         Map<21,21> m_Map{};
38         /*
39          * Брой изминали тикове
40          */
41         uint32_t m_Ticks{};
42     };
43 }
```

## **РасManClone/Player.cpp Файл Справка**

```
#include "Player.h"  
#include "WindowsFunctions.h"
```

## **РасManClone/Player.h Файл Справка**

```
#include "Point.h"  
#include "Map.h"  
#include <cstdlib>
```

### **Класове**

```
class РасMan::Player
```

### **Именни пространства**

```
namespace РасMan
```

## PacManClone/Player.h

Вижте документацията за този файл.1 #pragma once

```
2 #include "Point.h"
3 #include "Map.h"
4 #include <cstdlib>
5 namespace PacMan {
6     class Player {
7     public:
8         Player() = default;
9         Player(const Point startPosition) : position(startPosition) {}
10        void Move(Point dir) { position += dir; };
11        void Update(Map<21, 21>& map, Point ghostPositions[4]);
12
13        bool isDead{};
14        Point position{};
15        uint32_t direction{};
16        uint32_t score{};
17        uint32_t scoreMultiplier{};
18        uint32_t multiplierTicks{};
19        bool isPowered{};
20    };
21 }
```

## **РасManClone/Point.h Файл Справка**

```
#include <cstdint>
#include <cmath>
```

### **Класове**

```
class РасMan::Point
```

### **Именни пространства**

```
namespace РасMan
```

## PacManClone/Point.h

Вижте документацията за този файл.1 #pragma once

```
2 #include <stdint>
3 #include <cmath>
4 namespace PacMan {
5     class Point {
6     public:
10         Point() = default;
14         Point(float x, float y) : x(x), y(y) {}
18         Point(const Point&) = default;
19
20         Point& operator=(const Point& other) = default;
21         bool operator==(const Point& rhs) {
22             return (this->x == rhs.x && this->y == rhs.y);
23         }
24         Point& operator+=(const Point& rhs)
25         {
26             this->x += rhs.x;
27             this->y += rhs.y;
28             return *this;
29         }
30         Point& operator-=(const Point& rhs)
31         {
32             this->x -= rhs.x;
33             this->y -= rhs.y;
34             return *this;
35         }
36         bool operator!=(const Point& rhs)
37         {
38             return !(*this==rhs);
39         }
40         Point operator+(const Point& rhs)
41         {
42             float x = this->x + rhs.x;
43             float y = this->y + rhs.y;
44             return { x,y };
45         }
46         Point& operator*(float rhs) {
47             this->x *= rhs;
48             this->y *= rhs;
49             return *this;
50         }
51         Point GetTrunc() { return { trunc(floor(x)),trunc(floor(y)) }; };
52         float x{};
53         float y{};
54     };
55
56 }
```

## РacManClone/WindowsFunctions.cpp Файл Справка

```
#include "WindowsFunctions.h"
```

### Функции

```
void GotoXY (SHORT x, SHORT y)
```

```
POINT GetCursorPosition ()
```

```
void ClearScreen ()
```

```
void ShowConsoleCursor (bool showFlag)
```

---

### Функции Документация

#### **void ClearScreen ()**

Функция която трие екрана

#### **POINT GetCursorPosition ()**

Функция която взима позицията на курсора

#### **void GotoXY (SHORT x, SHORT y)**

Функция която казва на курсора да отиде на x,y

#### **void ShowConsoleCursor (bool showFlag)**

Функция която показва или скрива курсора

## РacManClone/WindowsFunctions.h Файл Справка

```
#include <Windows.h>
```

### Функции

void **GotoXY** (SHORT x, SHORT y)

POINT **GetCursorPosition** ()

void **ClearScreen** ()

void **ShowConsoleCursor** (bool showFlag)

---

### Функции Документация

#### void **ClearScreen** ()

Функция която трие екрана

#### POINT **GetCursorPosition** ()

Функция която взима позицията на курсора

#### void **GotoXY** (SHORT x, SHORT y)

Функция която казва на курсора да отиде на x,y

#### void **ShowConsoleCursor** (bool showFlag)

Функция която показва или скрива курсора



## PacManClone/WindowsFunctions.h

Вижте документацията за този файл.1 `#pragma once`  
2 `#include <Windows.h>`  
3  
7 `void GotoXY(SHORT x, SHORT y);`  
11 `POINT GetCursorPosition();`  
15 `void ClearScreen();`  
19 `void ShowConsoleCursor(bool showFlag);`

# Азбучен указател

INDEX