Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

Computer Programming

Console game – Snake.

Łukasz Kwiecień

Informatics, group 1, section 1

mgr inż. Krzysztof Pasterak

June 22, 2019

---

1 Topic

The Snake application is a simple console application based on popular "Snake" game. Program uses the Curses library. User can play the game, save his score and see top 20 results. Player controls snake on a bordered plane. Main goal is to earn as many points as possible. Game ends when snake runs into the screen border or itself.

2 Analysis and Development

The problem with creating console game comes down to two simple problems:
How to display something on the screen on the specific position?
How to clear the screen?

The curses library has functions that allows us to solve all of the problems previously described.

2.1 Algorithms and Data Structures

2.1.a) Algorithms

Program uses “Bubble Sort” sorting algorithm. Algorithm steps through the array, compares adjacent elements and swaps them if they are in wrong order. Bubble sort is used in ‘menu_func.c’ file in function “sort_scores’ to sort top 20 results(read from the file) , from the smallest to the largest.

```
for( int i = 0; i < 20; i++ ) //sorting using bubble sort
{
    for( int j = 0; j < 20 - 1; j++ )
    {
        if( scores[ j ] > scores[ j + 1 ] )
        {
            temp=scores[j];
            strcpy(temp1,names[j]);
            strcpy(names[j],names[j+1]);
            strcpy(names[j+1],temp1);
            scores[j]=scores[j+1];
            scores[j+1]=temp;
        }

    }
}
```

Above – implementation of bubble sort

2.1.b) Data Structures

All information about the snake's body is stored in doubly linked list.

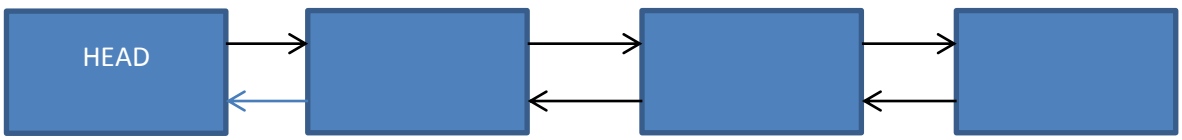
Structure looks like this:

```
/** doubly-linked list for storing informations about the snake */
struct coordinate
{
    int x; /**< x position */
    int y;/**< y position */
    int lifes;/**<number of lifes */
    int direction;/**< directon of movement */
    int length;/**< length of the snake */
    struct coordinate *next;/**< pointer to the next element of the list */
    struct coordinate *prev;/**< pointer to the previous element of the list */
};
```

Each element of the list has 2 pointers:

Next – pointer to the next element of the list

Prev – pointer to the previous element of the list



Doubly linked list - Visualization

3 External specification

User’s manual

3.1 Usage

The program requires the Curses library to be installed on the user’s computer.

User can choose one of the 3 options in main menu – start, top 20 and exit. Start runs the game function and allows player to play the snake. Top 20 shows top 20 results from the scores.txt file. Exit – quits the program.

Snake can move in 4 directions: up, down, left and right. It is controlled using the arrow keys.

3.2 Messages

The program prints warning when the player’s nickname length is incorrect.

4 Internal specification

Internal specification is moved to appendix.

5 Running and testing

The program has no memory leaks.

6 Conclusions

Creating a game is challenging but very interesting topic. Various problems can be solved in many different ways. Use of the Curses library was not necessary but, in comparison with standard libraries, game looks better and runs faster.

Appendix Description of types and functions

Snake

Generated by Doxygen 1.8.15

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 coordinate Struct Reference

`#include <structures.h>`

Public Attributes

- int x
- int y
- int lifes
- int direction
- int length
- struct coordinate * next
- struct coordinate * prev

3.1.1 Detailed Description

doubly-linked list for storing informations about the snake

3.1.2 Member Data Documentation

3.1.2.1 direction

`int coordinate::direction`

directon of movement

3.1.2.2 length

```
int coordinate::length
```

length of the snake

3.1.2.3 lifes

```
int coordinate::lifes
```

number of lifes

3.1.2.4 next

```
struct coordinate* coordinate::next
```

pointer to the next element of the list

3.1.2.5 prev

```
struct coordinate* coordinate::prev
```

pointer to the previous element of the list

3.1.2.6 x

```
int coordinate::x
```

x position

3.1.2.7 y

```
int coordinate::y
```

y position

The documentation for this struct was generated from the following file:

- structures.h

3.2 food_ Struct Reference

```
#include <structures.h>
```

Public Attributes

- int x
- int y
- int hit

3.2.1 Detailed Description

structure that holds data on food

3.2.2 Member Data Documentation

3.2.2.1 hit

```
int food_::hit
```

variable for checking if food has been eaten

3.2.2.2 x

```
int food_::x
```

x position

3.2.2.3 y

```
int food_::y
```

y position

The documentation for this struct was generated from the following file:

- structures.h

Chapter 4

File Documentation

4.1 map_func.h File Reference

Functions to create a map and generate food.

```
#include "structures.h"
```

Functions

- void draw_borders (WINDOW ∗screen)
- void generate_food (WINDOW ∗win, struct coordinate ∗head)

4.1.1 Detailed Description

Functions to create a map and generate food.

Author

Lukasz Kwiecien

4.1.2 Function Documentation

4.1.2.1 draw_borders()

```
void draw_borders (
            WINDOW * screen )
```

Function draws a map in a given window.

Parameters

screen	WINDOW∗ Pointer to the window(ncurses).

4.1.2.2 generate_food()

```
void generate_food (
            WINDOW * win,
            struct coordinate * head )
```

Function generates and prints food in a random position.

Parameters

win	WINDOW∗ Pointer to a window(ncurses).
head	struct coordinate∗ Pointer to the first element of the list

4.2 menu_func.h File Reference

Contains menu function and functions that run the game.

Functions

- void menu ()
- void game ()
- void dead (struct coordinate ∗head)
- void sort_scores (char name[ ], int score)
- void scores ()

4.2.1 Detailed Description

Contains menu function and functions that run the game.

Author

Lukasz Kwiecien

4.2.2 Function Documentation

4.2.2.1 dead()

```
void dead (
            struct coordinate * head )
```

Function displays a score and saves the nickname of the player when the game is over.

Parameters

head	struct coordinate* Pointer to the first element of the list

4.2.2.2 game()

```
void game ( )
```

Main game function, calls other necessary functions.

4.2.2.3 menu()

```
void menu ( )
```

Function displays menu and calls other functions depending on user's choice.

4.2.2.4 scores()

```
void scores ( )
```

Function displays top 20 results.

4.2.2.5 sort_scores()

```
void sort_scores (
            char name[],
            int score )
```

Function sorts the top 20 results achieved by players and writes them down in a text file.

Parameters

name[]	char
score	int

4.3 snake_move.h File Reference

Functions related to action of moving a snake.

```
#include "structures.h"
```

Functions

- int kbhit (void)
- void get_dir (struct coordinate *head)
- void bend (struct coordinate **head)
- void move_snake (WINDOW *win, WINDOW *field, struct coordinate *head)
- void collision (struct coordinate *head)

4.3.1 Detailed Description

Functions related to action of moving a snake.

Author

Lukasz Kwiecien

4.3.2 Function Documentation

4.3.2.1 bend()

```
void bend (
            struct coordinate ** head )
```

Function sets a correct direction of movement and position(x,y) for each element of a snake.

Parameters

head	struct coordinate** Pointer to a pointer to the first element of the list

4.3.2.2 collision()

```
void collision (
            struct coordinate * head )
```

Function detects collision between head of a snake and its body.

Parameters

head	struct coordinate* Pointer to the first element of the list

4.3.2.3 get_dir()

```
void get_dir (
            struct coordinate * head )
```

Function sets a position and direction of movement of the head of the snake, depending on the pressed key.

Parameters

head	struct coordinate* Pointer to the first element of the list

4.3.2.4 kbhit()

```
int kbhit (
            void  )
```

Function determines if a key has been pressed or not.

Parameters

void	

Returns

int returns 1 if key was pressed otherwise returns 0

4.3.2.5 move_snake()

```
void move_snake (
            WINDOW * win,
            WINDOW * field,
            struct coordinate * head )
```

Function prints all elements of a snake's body in the appropriate positions.

Parameters

win	WINDOW* Pointer to the window(ncurses).
field	WINDOW* Pointer to the window(ncurses).
head	struct coordinate* Pointer to the first element of the list

4.4 structures.h File Reference

Contains doubly linked list and its functions and structure that holds data on food.

Classes

- struct coordinate
- struct food_

Functions

- void append (struct coordinate **head)
- void free_mem (struct coordinate **head)

Variables

- struct coordinate * **head**
- struct food_ **food**

4.4.1 Detailed Description

Contains doubly linked list and its functions and structure that holds data on food.

Author

Lukasz Kwiecien

4.4.2 Function Documentation

4.4.2.1 append()

```
void append (
            struct coordinate ** head )
```

Function adds new element at the end of the doubly linked list.

Parameters

head	struct coordinate** Pointer to a pointer to the first element of the list

4.4.2.2 free_mem()

```
void free_mem (
            struct coordinate ** head )
```

Function removes all elements of the list.

Parameters

head	struct coordinate** Pointer to a pointer to the first element of the list