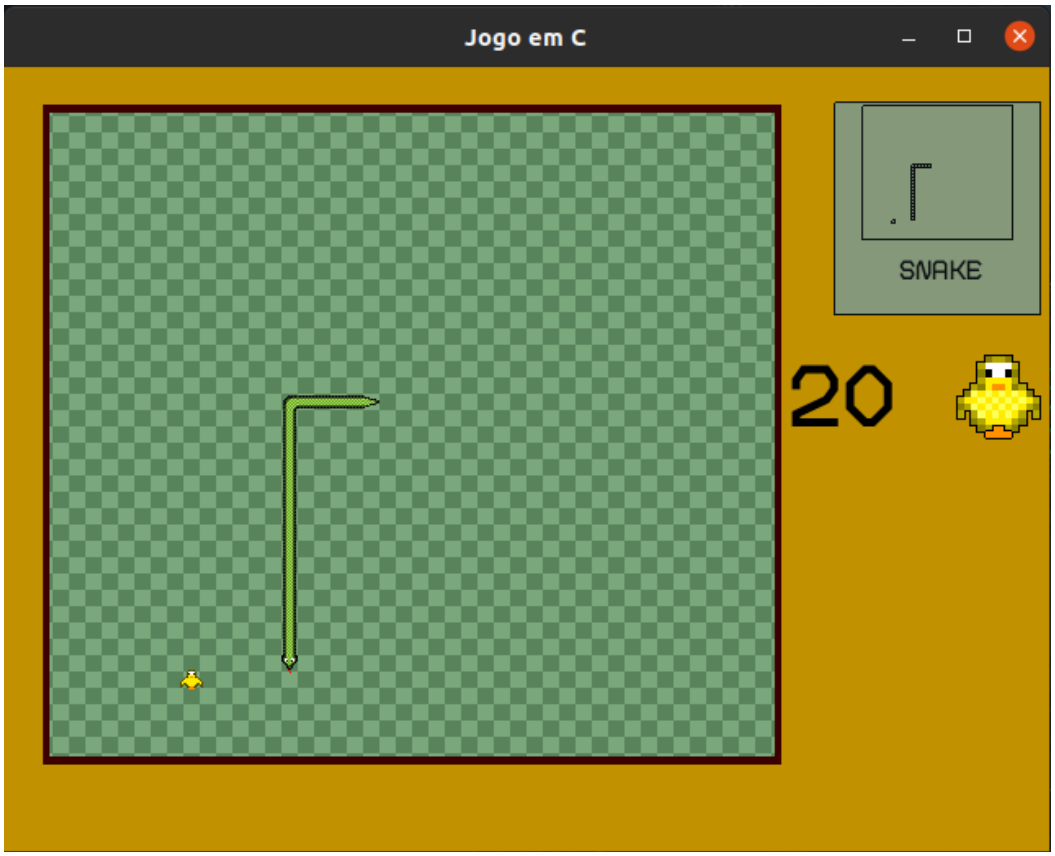
Snake



Snake foi o jogo proposto para o Trabalho prático da disciplina de Algoritmos e Estrutura de Dados do curso de Ciência da computação da UFSJ. O código do jogo foi dividido em três módulos principais: *cobra.h*, *telas.h* e *desenhos.h*.

Módulos

- *cobra.h*

No módulo *cobra.h* estão as *structs* usadas no programa, são elas:
- “segmento”:

```
typedef struct{
    int posix;
    int posiy;
}segmento;
```

- “cobra”:

```
typedef struct{
    int tamanho;
    int direcao;
    segmento *segmentos;
}cobra;
```

- “fruta”:

```
typedef struct{
    int posix;
    int posiy;
}fruta;
```

Nesse módulo ainda estão as funções relacionadas à cobra, que são:

```
cobra *gera_cobra();
// Aloca a cobra dinamicamente

void destroi_cobra(cobra *elvira);
// Libera a memória alocada

void dificuldade(int *cont, int lvl, cobra *elvira, fruta *maca, int *pontuacao, int *tela);
// Seta a dificuldade do jogo.


fruta gera_fruta(cobra *elvira);
// Gera a maçã em um lugar aleatório toda vez que é comida;


void andar(cobra *elvira, fruta *maca, int *pontuacao, int *tela);
// Anda com a cobra:
//      - Copia a posição da cabeça para a última posição do vetor;
//      - Soma o tamanho do segmento em px na direção do movimento;
//      - Chama @organiza_vetor().


void organiza_vetor(cobra *elvira);
// Organiza o vetor de posições dos segmentos da cobra:
//      - Salva o valor da ultima posição do vetor;
//      - Copia todas os valores para a posição seguinte;
//      - Copia o valor salvo para a posição zero.
```

As funções *gera_cobra()* e *destroi_cobra()* são auto-explicativas. A função *dificuldade()* é chamada pela main em toda iteração do loop principal, e chama a função andar quando a variável *cont*, que é incrementada a cada iteração do loop principal, for divisível pelo valor setado na variável *lvl*. A função *gera_fruta()* é chamada toda vez que a fruta é comida pela cobra, ela retorna uma *fruta* com uma posição x/y gerada aleatoriamente em alguma posição da tela não ocupada pela cobra. As funções *andar()* e *organiza_vetor()* serão vistas a seguir:

Movimentação

A movimentação da cobra é realizada em sua maior parte pela função *andar()* do módulo *cobra.h*, essa função recebe como parâmetros a cobra, a fruta, a pontuação e a tela (que será explicada mais à frente). Ao ser chamada, ela confere a direção para a qual a cobra está andando para decidir em qual atributo da coordenada da cobra ela vai realizar a operação. Para exemplo vamos pegar o caso onde a direção da cobra é “PARA_DIREITA”, nesse caso é necessário somar 10 à coordenada x da cabeça, e o restante dos segmentos assumem o valor do segmento anterior. Para isso a função *andar()* copia os valores de x e y da cabeça para o último segmento da cobra somando 10 na coordenada x, após ter esses valores copiados é chamada a função *organiza_vetor()* que coloca o último elemento do vetor de coordenadas da cobra, que agora é a cabeça, na posição zero do vetor e os outros elementos andam uma posição para frente.

(100, 100)	(90, 100)	(80, 100)	(70, 100)	(60, 100)

↓ *andar()*

(100, 100)	(90, 100)	(80, 100)	(70, 100)	(110, 100)

↓ *organiza_vetor()*

(110, 100)	(100, 100)	(90, 100)	(80, 100)	(70, 100)

Após organizar o vetor a função *andar()* faz o uso de mais duas funções implementadas em *cobra.c* para fazer as verificações necessárias. Primeiro é chamada a função *colidiu()*, que retorna verdadeiro caso a cobra tenha colidido com a parede ou com ela mesma, logo após é chamada a função *comeu()*, que retorna verdadeiro caso a cobra tenha comido a fruta (as implementações de *comeu()* e *colidiu()* serão apresentadas a seguir). Pronto, depois de todo esse processo realizado a cobra moveu uma casa e está pronta para se mover de novo.

Colisão

A colisão da cobra é feita por duas funções diferentes, uma para verificar a colisão dela com ela mesmo ou com os limites do mapa, o que resulta no fim do jogo, e outra para verificar a colisão dela com a fruta, que resulta no crescimento dela e na pontuação do jogador. A primeira função é a *comeu()*, que retorna 1 caso ela tenha comido e 0 caso contrário, essa função verifica se as coordenadas da cabeça são iguais a coordenada da fruta, se forem iguais significa que ela comeu e pode crescer. A segunda função é a *colidiu()*, que verifica a colisão da cobra em duas etapas. Na primeira etapa ela confere se a cobra colidiu com ela mesma ao realizar a movimentação, pra isso ela confere se a coordenada da cabeça é igual à coordenada de algum dos segmentos do corpo dela. Já na segunda etapa ela confere se a cabeça da cobra passou dos limites da tela, acontecendo qualquer um dos casos, tanto da primeira etapa quanto da segunda, a função *colidiu()* retorna 1 e encerra sua execução, caso contrário ela retorna 0.

Tendo agora como a função *comeu()* e todas as funções chamadas por ela funcionam, pode-se conferir sua implementação a seguir:

```
void andar(cobra *elvira, fruta *maca, int *pontuacao, int *tela){
   switch (elvira->direcao){
   case PARA_DIREITA:
       (elvira->segmentos)[(elvira->tamanho)-1].posix =
(elvira->segmentos)[0].posix + TAMANHO_SEGMENTO;
       (elvira->segmentos)[(elvira->tamanho)-1].posiy =
(elvira->segmentos)[0].posiy;
       organiza_vetor(elvira);
       if(colidiu(elvira)){
           *tela = GAME_OVER;
       };
       if(comeu(elvira, maca)){
           (*pontuacao)++;
           cresce(elvira);
           *maca = gera_fruta(elvira);
       }
       break;
    }
}
```

No exemplo acima foi apresentado apenas o *case PARA_DIREITA*, nos outros casos o que muda é onde somamos ou subtraímos 10 unidades:

- *PARA_ESQUERDA*: x-10
- *PARA_CIMA*: y-10
- *PARA_BAIXO*: y+10

Movimentação e colisão concluídas, vamos para veremos agora as telas.

- *"telas.h"*

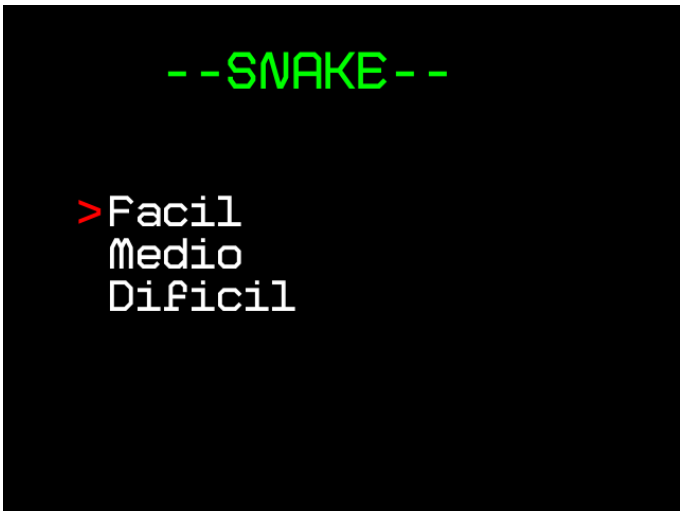
O módulo *telas.h* é responsável por controlar qual a tela está sendo apresentada ao jogador. Nele estão presentes as seguintes funções:

```
void game_over(cobra *elvira, int *tela, int *pontuacao, char *apelido);

void escolhe_dificuldade(int *tela, int *dif);

void menu_iniciar(int *tela);

void recebe_nick(int *tela, char apelido[4]);

void jogar(cobra *elvira, int lvl, int *cont, fruta *maca, int *pontuacao, int *tela);

void ranking(int *tela);
```

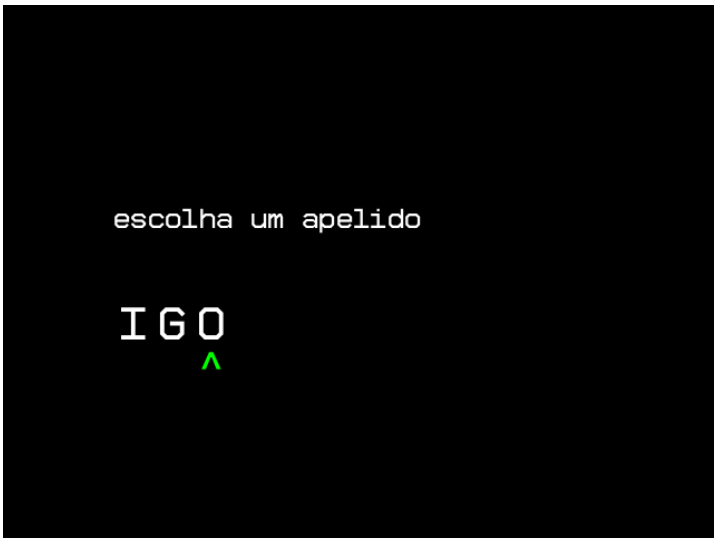
Cada uma dessas funções é responsável por desenhar a tela correspondente para o jogador. Para que isso seja possível, uma variável do *int tela* foi criada na *main()* e cada uma dessas funções é chamada quando essa variável tem o valor correspondente a ela.

Quando iniciado, a variável tela está definida como *MENU_INICIAR*, então o menu iniciar é a primeira tela apresentada ao usuário:

Nessa tela é possível usar as setinhas para baixo e para cima para navegar entre as opções e *enter* para selecionar a opção desejada. Ao selecionar “Iniciar”, a variável tela é alterada de *MENU_INICIAR* para *ESCOLHE_DIFICULDADE*, então a tela de dificuldade é mostrada ao jogador:



Nessa tela é possível usar as setinhas para baixo e para cima para navegar entre as opções e *space* para selecionar a dificuldade desejada. Ao selecionar a dificuldade, a variável tela é alterada de *ESCOLHE_DIFICULDADE* para *NICK_NAME*, então a tela de apelido é mostrada ao jogador:



Nessa tela é possível usar as setinhas para a direita ou para a esquerda para poder selecionar a letra que quer alterar, e as setinhas para cima e para baixo para alterar a letra escolhida, montando assim seu apelido com 3 letras. Após escolhido o

apelido, o jogador pode apertar *enter* para iniciar o jogo. As demais funções do módulo serão vistas a seguir:

- ***jogar()***

A função *jogar()* é chamada após o jogador escolher o apelido, nela são desenhadas todas as *sprites* em suas respectivas coordenadas. É nessa função também que ocorre a conferência caso alguma tecla seja pressionada para mudar a direção da cobra.

- ***ranking()***

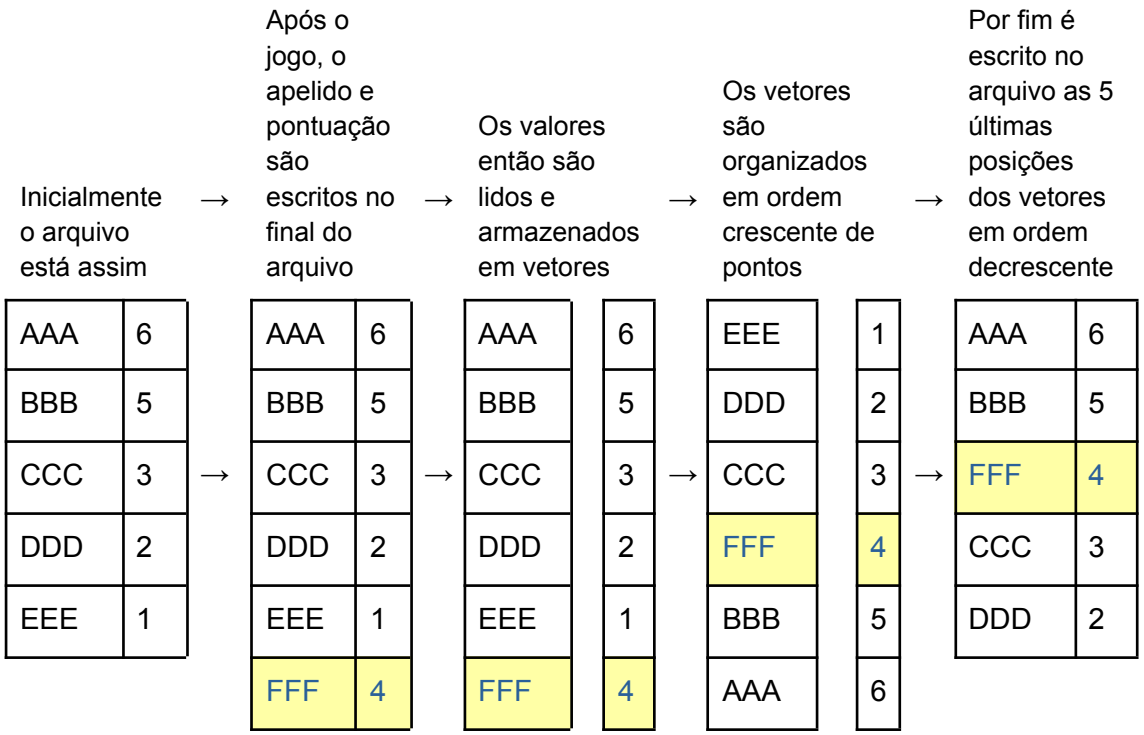
A função *ranking()* é responsável por ler o arquivo *ranking.csv* e imprimir na tela as informações contidas nele, esse arquivo contém as 5 melhores pontuações já registradas no jogo. Esse arquivo é atualizado toda vez que a função *game_over()* é chamada.

- ***game_over()***

A função *game_over()* é chamada quando a cobra colide com ela mesma ou com os limites da tela. essa função é responsável por imprimir a tela de game over para o jogar, porém, antes de imprimir, é realizada a classificação do jogador, para isso é chamada a função *rankear()*, presente em *telas.c*, que realiza os seguintes passos:

1. Abre o arquivo *ranking.csv* no modo de inserção (“a”), imprime o apelido e a pontuação ao final do arquivo e fecha ele.
2. Abre novamente o arquivo, porém no modo leitura (“r”), então coloca os apelidos em um vetor de strings e as pontuações em um vetor de inteiros, e por fim fecha o arquivo.
3. Realiza a ordenação do vetor de pontos usando um algoritmo de ordenação, e replica toda e qualquer alteração feita neste vetor também ao vetor de apelidos.

4. Por fim abre novamente o arquivo agora em modo de escrita e imprime no arquivo as 5 últimas posições do vetor em ordem decrescente.



Finalizado o módulo *telas.h*, veremos agora o último módulo.

- *"desenhos.h"*

O módulo *desenhos.h* é o responsável por desenhar a cobra usando a sprite certa para cada segmento. Nele estão contidas três funções, são elas:

```
void desenha_cauda(cobra *elvira);

void desenha_segmento(cobra *elvira, int posicao);

void desenha_cabeca(cobra *elvira);
```

A função *desenha_cauda()* é responsável por desenhar o segmento de índice (tamanho - 1) da cobra, para isso ela analisa a posição do segmento de índice (tamanho - 2) em relação a ele mesmo para decidir qual sprite deve ser desenhada, por exemplo, caso o segmento[tam-2] esteja acima do segmento[tam-1] então a sprite que deverá ser desenhada é a "cauda_cima.gif", pois a cauda está subindo.

A função *desenha_segmento()* é responsável por desenhar todos os segmentos da cobra, analisando a posição do anterior e do posterior para decidir qual sprite usar.

Por fim, a função *desenha_cabeça()* é responsável por desenhar o segmento de índice 0, para isso ela analisa o segmento de índice 1 para saber qual sprite deve ser desenhada.

Sprites

Sprites utilizadas no jogo:

Cobra:

Fruta (que é um pintinho):

Background: