Entrega 1 (02/06)

Na nossa simulação, o usuário terá acesso aos resultados da simulação, ou seja, horários em que cada emergência foi requisitada, em qual bairro, etc;

Além disso, poderá procurar por status de ocorrências, etc (coisas das próximas entregas);

Organização dos códigos:

- Structs: usar typedef;
- Nome de função: snake case;
- Nome de struct: Primeira letra maiúscula;
- Nome de constantes: todas as letras maiúsculas;
- Comentário no início de cada arquivo com uma pequena descrição do seu propósito;
- Comentário antes de cada função, explicando brevemente o que ela faz;
- Comentar parte do código que nós achemos complexa para o outro entender;
- CHAVES: Explícita e ao lado
- Fazer espaçado o que der para ficar mais clara a leitura (ao invés de if(x==1) fazer if (x == 1), ao invés de a+b=3, fazer a + b = 3, etc);
-

Fazer primeira entrega:

- Bairro: Struct, Hashing, Inserir os bairros na estrutura; - Igor
- Policia: Struct, Hashing, Inserir as polícias na estrutura; - Milena
- Hospital: Struct, Hashing, Inserir os hospitais na estrutura; - Milena
- Bombeiro: Struct, Hashing, Inserir os bombeiros na estrutura; - Milena
- Ocorrências: Struct, criar as funções randômicas para gerar os registros; - Igor
- Fila: Fila, fazer as funções de de inserção, remoção, etc, pois essa será dinâmica, ao passo em que novas ocorrências surgem; - Igor
- Definição do tempo: através de ciclos, ainda não sabemos exatamente quantas ocorrências por ciclo;

Nossa tabela Hashing de será perfeita, ou seja, as chaves nunca irão colidir, uma vez que elas já são conhecidas e, consequentemente, suas posições também. Utilizaremos a divisão para fazer o cálculo da posição em que os dados ficarão.

Funcionalidades:

- Cadastro de bairros (Tabela Hashing). -> estático (Quantidade de bairros e seus respectivos nomes não serão definidos pelo usuário)

Como será estático, já vamos definir a quantidade de bairros e suas “características”

Bairro:

quantidade -> 9 //A princípio

struct:

char nome[50];

int ID;

Lista dos bairros:

1) Jardim das Anas
2) Vila São Pedroso
3) Gugarujá
4) Igornema
5) Miicca
6) Nova Olyans
7) Fenda dos Parafusos
8) Ciriacabana
9) Vila Santa Caroline

Estrutura utilizada para armazenar: Hashing

- Cadastro de unidades de serviço (hospital, bombeiros, etc.). -> Na **Fase 1**, o "Cadastro de unidades de serviço" envolverá tanto a definição da estrutura para representar essas unidades quanto a implementação da estrutura de dados escolhida para armazená-las.

  Aqui está o que isso significa:

  1. **Definição da Estrutura:** Você precisará decidir quais informações cada unidade de serviço irá conter (por exemplo, ID, tipo, talvez um nome específico). Isso define a "estrutura" de cada unidade.

  2. **Implementação da Estrutura de Dados:** Você precisará escolher uma forma de armazenar múltiplas dessas unidades. Embora o uso de Hashing seja explicitamente mencionado

Para ver se o serviço está disponível, utilizaremos o tipo bool da biblioteca <stdbool.h>, para tornar o código mais legível;

SERVIÇOS:

- Polícia

  quantidade -> 3

  struct:

int ID;

int qtd_viaturas;

int viaturas_disp;

bool disponivel;

Lista da polícia:

1) Nome1
2) Nome2
3) Nome3

Estrutura utilizada para armazenar: Hashing

- Hospital (92)

quantidade -> 2

struct:

int ID;

int qtd_ambulancias;

int ambulancias_disp;

bool disponivel;

Lista dos hospitais:

1) Nome1
2) Nome2

Estrutura utilizada para armazenar: Hashing

- Bombeiro (93)

quantidade -> 2

struct:

int ID;

int qtd_caminhoes;

int caminhoes_disp;

bool disponivel;

Lista dos bombeiros:

1) Nome1
2) Nome2

Estrutura utilizada para armazenar: Hashing

- Simulação de filas de atendimento. -> Na primeira entrega, faremos uma fila só;

Struct ocorrencias:

int ID;

char tipo[1000];

int servicos_requisitados[3];

*pessoa morador;

char bairro;

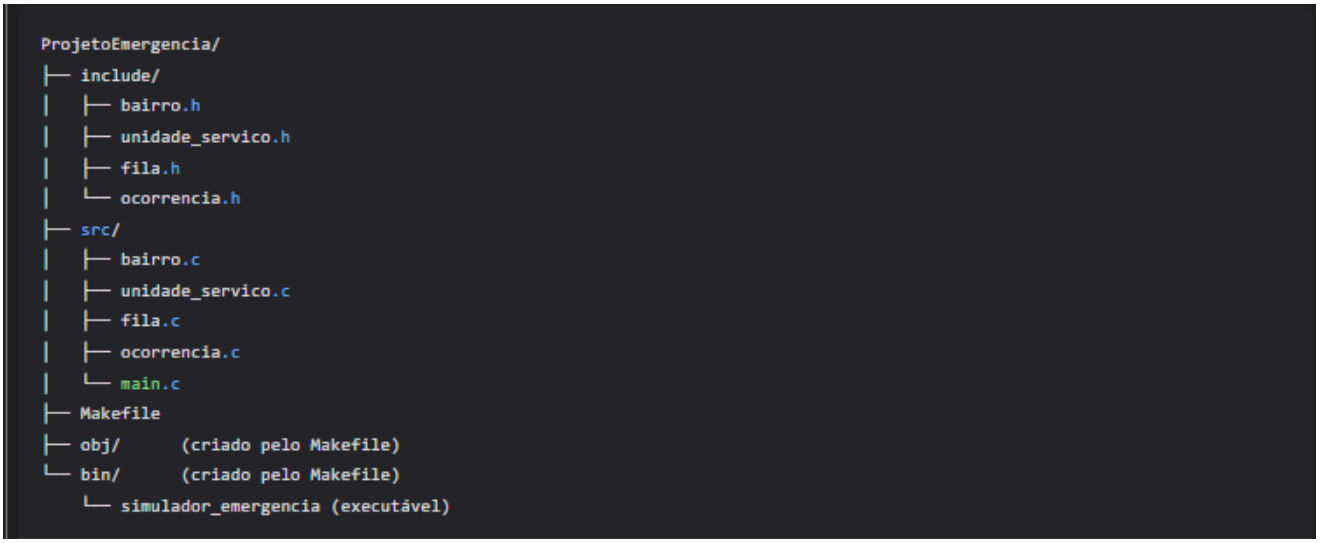
Struct pessoa:

int id

char nome[x];

char bairro;

- Organização da ordem de chegada de ocorrências (definir unidades de tempo). -> **Como vamos definir o tempo? Vão chegar novas ocorrências a cada quanto tempo?**



-> Não precisa do Makefile pois o CodeBlocks já faz esse trabalho.