Trabalho de Estrutura de Dados

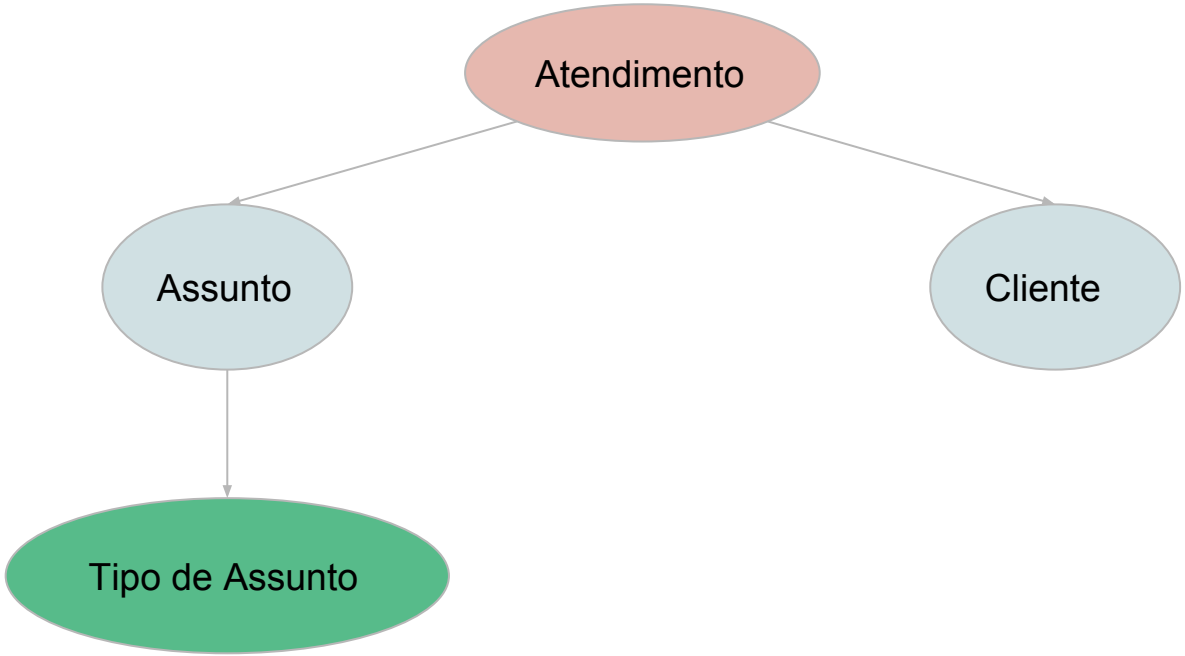
Jordan Jusbig Salas Cuno
Jose Luis Huillca Mango

Classes criadas

- class Cliente()
  - cpf
  - nome
  - idade => rand()
- class TipoAssunto()
  - tipo
  - titulo
  - urgencia => rand(0-10)

- class Assunto()
  - tipo => TipoAssunto()
  - descricao
  - providencias
  - duracaoAtendimento
- class Atendimento()
  - cliente => Cliente()
  - assuntos => MyList<Assunto()>
  - horaLlegada
  - horaAtendimento
  - prioridade => media{...,...,...}

Classes criadas



Estruturas

- MyList<T>() => lista encadeada
  - void inserir() => O(1)
  - void excluir() => O(n)

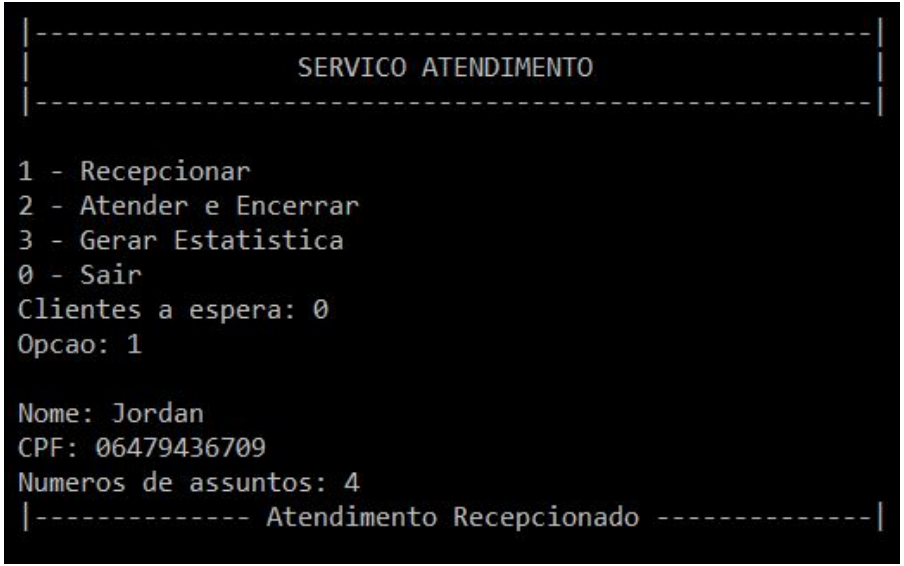
- MyHeap() => armazenar Atendimento()
  - void inserir() => O(log n)
  - Atendimento excluir() => O(log n)
  - get maior prioridade O(1)
  - void enfileirar() => O(1)
  - void crearHeap() => O(n*log n)

- MyHash() => armazenar No*
  - key = tipo de assunto
  - value = No(){
    - contador
    - duracaoAcumulada
    - tipo de assunto
    - }
  - get média de value [k] é O(1)

Funções

Primeiro criamos uma lista de tipos aleatórios de assuntos.

1. **recepcionar**() : Adicionamos um objeto *Atendimento*, onde a idade do *Cliente* é aleatória, só inserimos um número que é a quantidade de assuntos que vai ser gerados aleatoriamente. O objeto será armazenado em um heap, onde no início o heap não será ordenado por prioridade, se não ele vai trabalhar como uma lista.



Funções

Recepcionar:

- Lista de tipo de assunto é um array dinâmico *listaTipoAssunto*

```
8    // Create list listaTipoAssunto
9    listaTipoAssunto = new TipoAssunto[MAX_SIZE_LIST];
10   for(int i=0; i<MAX_SIZE_LIST; i++){
11     ostringstream str1;
12     str1<<i+1;
13     int urgenciaRand = rand() % GRAU_URGENCIA + 1;
14     TipoAssunto tipoAsuntoTemporal(i+1, "Asunto "+str1.str(), urgenciaRand);
15     listaTipoAssunto[i] = tipoAsuntoTemporal;
16   }
```

- O heap de atendimentos inseridos é *heapAtendimento*

```
79 void servicioAtendimento::recepcionar(Cliente cliente_, MyList<Assunto> listaAssunto_)
80 {
81   time_t tempoAgora;
82   time(&tempoAgora);
83
84   Atendimento Objeto_Atendimento(cliente_, listaAssunto_, tempoAgora);
85   heapAtendimento.enfilerar(Objeto_Atendimento);
86 }
```

Funções

- **atender**() : Precisamos calcular a prioridade de cada um dos *Atendimentos* que estão em nosso heap, e logo criar nosso heap baseado nas prioridades. Depois que o heap é criado, tiramos o *Atendimento* com a maior prioridade.

Funções

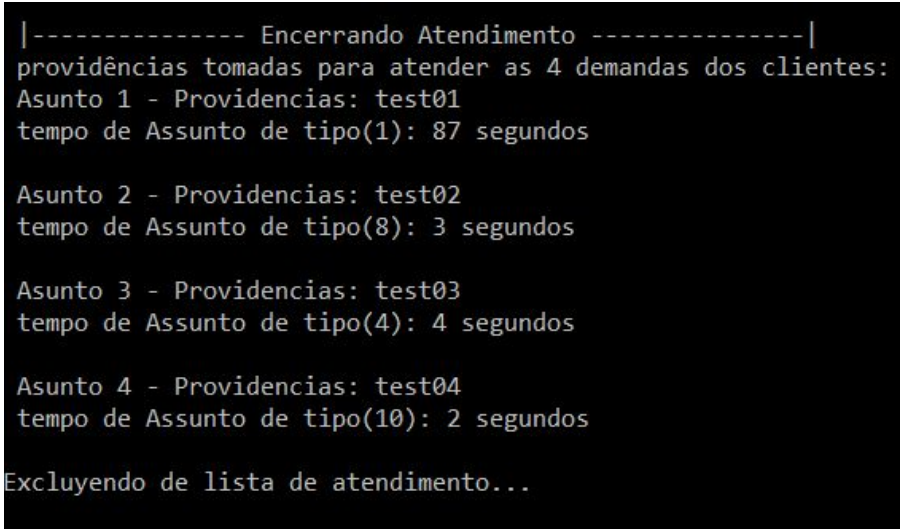
Atender:

- Atualizar as prioridades dos *Atendimentos* do *heapAtendimento*

```
101  Atendimento servicioAtendimento::atender()
102  {
103    time_t tempoAgora;
104    time(&tempoAgora);
105    int tamanhoLista = heapAtendimento.getTamanho();
106    for(int i=0; i<tamanhoLista; i++){
107        heapAtendimento.setAtendimento(i, tempoAgora);
108    }
109    heapAtendimento.crearHeap();
110
111    // Get max prioridade
112    Atendimento atendimentoAtender = heapAtendimento.excluir();
113    return atendimentoAtender;
114  }
```

Funções

- **encerrar**(): Do *Atendimento* tirado em atender(), inserimos as providências de cada *Assunto* e modificamos sua duração de atendimento pelo tipo de assunto, então colocamos no hash um nó, onde tem uma chave que é o tipo de assunto e um valor que é a duração de atendimento acumulada pelo tipo de assunto.



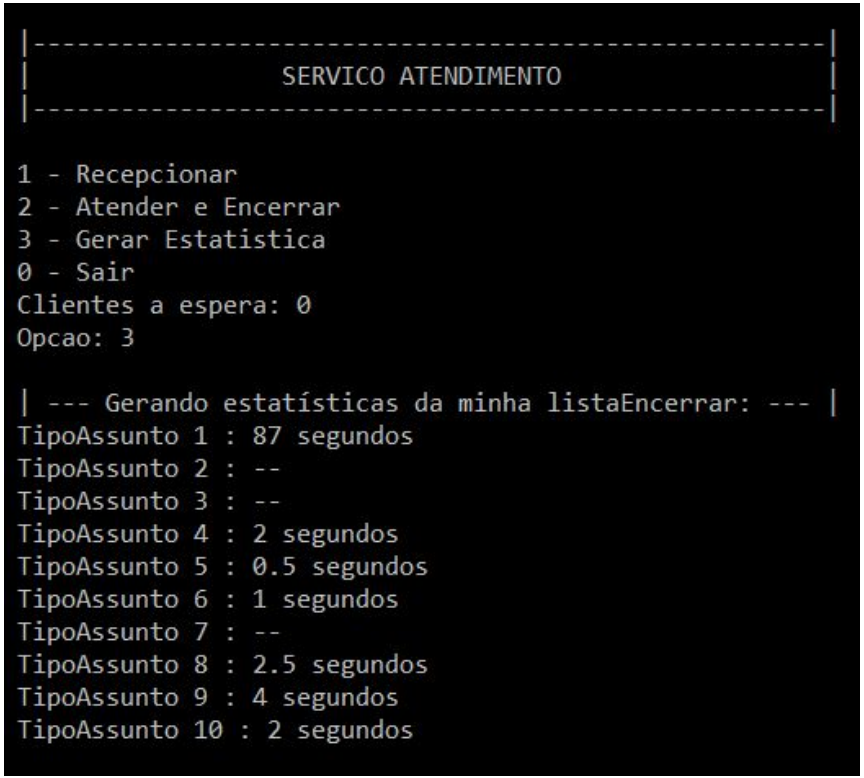
Funções

Encerrar:

```
39      ostringstream str0;
40      str0<<num_assuntos;
41      cout<<" providências tomadas para atender as "<< str0.str() <<" demandas dos clientes: "<<endl;
42      for(int i=0; i<num_assuntos; i++){
43          ostringstream str1;
44          str1<<i+1;
45          cout<<" Asunto " + str1.str()<<" - Providencias: ";
46          cin>>aux_providencia;
47
48          Assunto assunto = atendimento.getAssuntos().get(i);
49          assunto.setProvidencias(aux_providencia);
50          time(&tempoProvidencias);
51          double duracaoAtendimento = tempoProvidencias-horaInicio;
52          ostringstream str2;
53          str2<<assunto.getTipo().getTipo();
54          cout<<" tempo de Assunto de tipo("<<str2.str()<<"): "<<duracaoAtendimento<<" segundos"<<endl<<endl;
55          assunto.setDuracaoAtendimento(duracaoAtendimento);
56          horaInicio += duracaoAtendimento;
57          listaEncerrar.inserir(assunto.getTipo().getTipo(), assunto.getDuracaoAtendimento());
58      }
```

Funções

- **gerarEstadistica**(): Nós geramos as estatísticas com os dados que estão em nosso hash, pegamos todos esses dados como se fossem do dia atual. A chave do nosso hash é do *tipo* de *Assunto* "k", que vai armazenar uma estrutura Nó, onde ele tem a soma acumulada das durações de atenção e a quantidade desse tipo de assunto que está nesse hash com a chave "k" .



- Mostramos todos os tipos de assuntos com suas respectivas médias (duração da atenção cumulativa entre quantidade).

Gerar Estatística:

```
62  void servicioAtendimento::gerarEstatistica()
63  {
64      cout<<" | --- Gerando estatísticas da minha listaEncerrar: --- |"<<endl;
65      for(int i=1; i<=listaEncerrar.getTamanho();i++){
66          ostringstream str1;
67          str1<<i;
68          float mediaAtendimento = 0.0;
69          if(!listaEncerrar.estaVazio(i)){
70              mediaAtendimento = listaEncerrar.getMedia(i);
71              cout<<" TipoAssunto " + str1.str() + " : "<<mediaAtendimento<< " segundos" <<endl;
72          }else{
73              cout<<" TipoAssunto " + str1.str() + " : -- " <<endl;
74          }
75
76      }
77  }
```

Obrigado!