UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Campus CAMPO MOURÃO

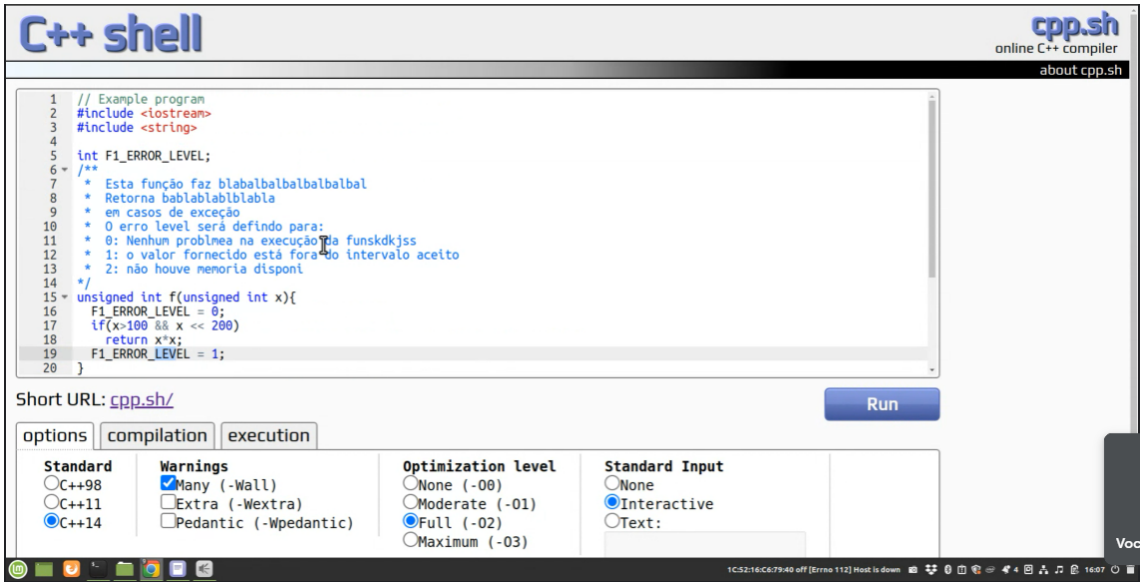
APOO

RELATÓRIO DE AULA

26/10/2021: Associação: Composição e Agregação. Construtores e Destrutores.

Estudante: Reginaldo Gregório de Souza Neto RA: 2252813

ERROR LEVEL



O professor explicou como era a estratégia para as exceções de erros nos primórdios da computação (1970 – 1980), onde os programadores utilizavam com frequência uma variável int para definir um número específico referente a cada tipo de erro.

Código:

```
// Example program
#include <iostream>
#include <string>

/**
 * Esta função faz blabalbalbalbalbalbal
 * Retorna bablablablblabla
 * em casos de exceção
 * O erro level será defindo para:
 * 0: Nenhum problmea na execução da funskdkjss
 * 1: o valor fornecido está fora do intervalo aceito
 * 2: não houve memoria disponi
*/
int F1_ERROR_LEVEL;
unsigned int f(unsigned int x){
 F1_ERROR_LEVEL = 0;
 if(x>100 && x << 200)
  return x*x;
 F1_ERROR_LEVEL = 1;
}


  int main()
  {
   unsigned int result = f(2);
   if(!F1_ERROR_LEVEL)
     std::cout << "Hello, " << f(2) << "!\n";
   else
     std::cout << "Erro com a função código " <<
  F1_ERROR_LEVEL << "!\n";
  }
```

TRHOW, TRY & CATCH

Quando ocorre algum erro o algoritmo “lança” (trhow) algum objeto específico. Ao utilizar o try, o compilador tenta executar o código dentro dos escopos do try, caso alguma chamada arremesse alguma coisa, ele analisa qual o tipo do objeto arremessado e vai em busca de um catch que possa “catá-lo”. Caso não haja nenhum catch com a tipagem do arremesso, ele segue “atravessando” até que alguém o pare, ou então ele volta à quem chamou a função main. É possível lançar objetos complexos complexos.

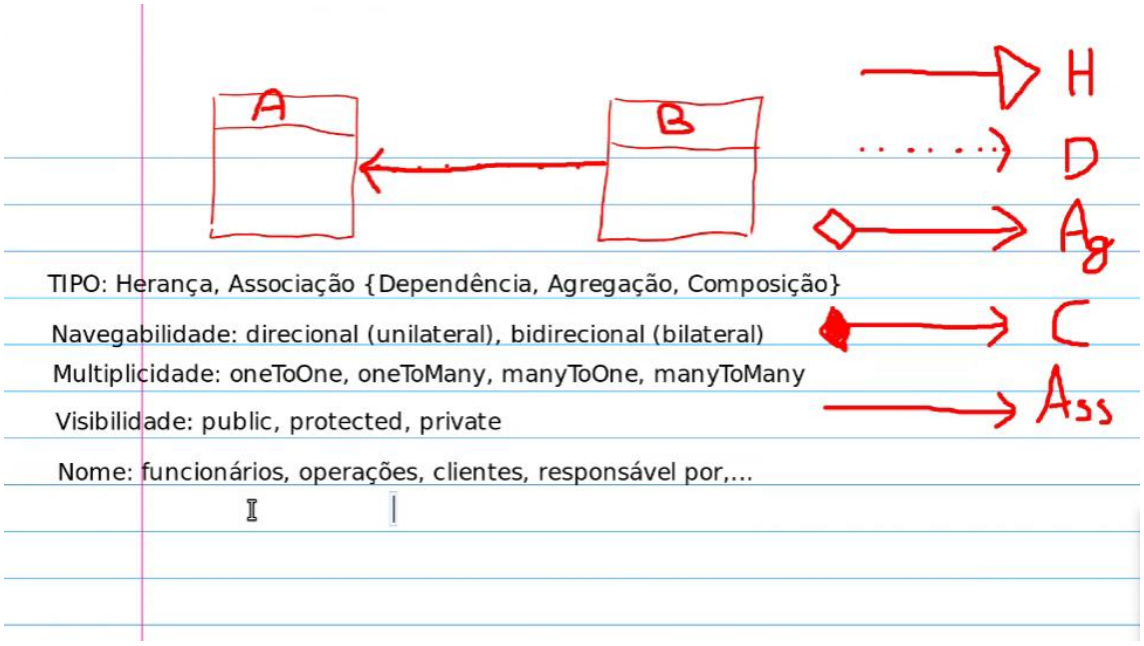
```
// Example program
#include <iostream>
#include <string>

class Range{
   public:
    int start, end, value;
     Range(int start, int end, int value){ this->start =
start; this->end = end; this->value = value;}
};

unsigned int f(unsigned int x){
 if(x>100 && x < 200)
   return x*x;
 throw Range(101, 199, x);
}

int g(int x){
 return f(x*2);
}

int main()
{
  try{
    unsigned int result = g(2);
    std::cout << "Hello, " << result << "!\n";
  }
  catch(Range erro){ std::cout << "Erro de intervalo
com a função f: start= " << erro.start << "end= " <<
erro.end << " valor passado= " << erro.value << "!\n"; }
}
```

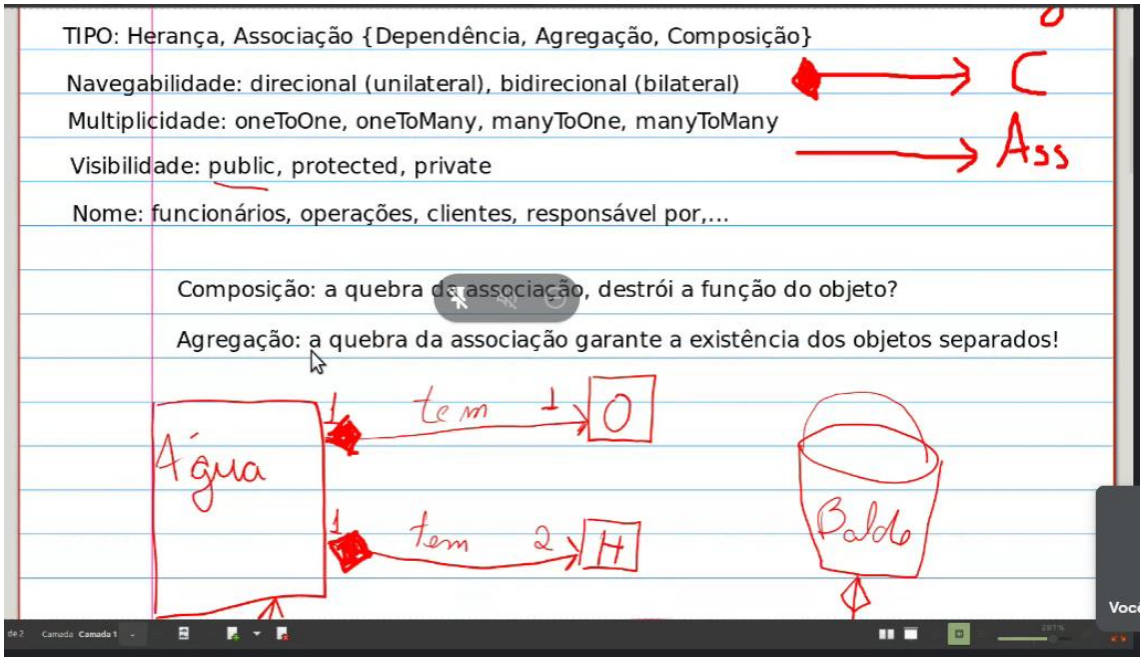
HERANÇA

O professor retomou o conceito a respeito de herança explicitando que os objetos são “agrupados” por suas características (atributos) semelhantes. Validando deste modo a capacidade e relevância da herança que através dela é possível fazer com que objetos semelhantes possuam métodos e atributos semelhantes de um mesmo “pai”.

Junto a isso ele retomou os conceitos da lista de herança e sugeriu que a lista, fila e pilha pudessem ser herdadas de uma classe chamada coleção. Assim como é feito em Java. Documentação disponível: https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html

ASSOCIAÇÃO





PARA CASA:

Entregar a tarefa 7 - Resenhas dos capítulos 2 e 9 (Fowler)

Entrega dos casos de uso do trabalho.