UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

CAMPUS ZONA OESTE UERJ

FABIO AUGUSTO DA SILVA MAGALHAES

GUSTAVO DE OLIVEIRA EUFRAZIO

Teste Unitário

Teste de Software

RIO DE JANEIRO

2025

GUSTAVO DE OLIVEIRA EUFRAZIO – 202420561811

FABIO AUGUSTO DA SILVA MAGALHAES – 202420526411

Introdução:

Esta documentação é referente a implementação de uma solução para o seguinte problema: Algoritmo para calcular o preço de ingressos de um zoológico. Para implementação do teste unitário foi utilizada a ferramenta Minunit framework de teste mínimo para C.

Explicando o código:

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include "funcoes.h"
#include "minunit-master/minunit.h"
```

Acima estão as bibliotecas além do arquivo onde contém as funções “funcoes.h” e o framework Minunit conforme a referência a localização do arquivo “minunit.h”.

```c
MU_TEST(test_calculaIngressos){
    int caso1[] = {5, 10, 12};      // todas crianças → 3*10 = 30
    int caso2[] = {20, 30};         // adultos → 2*30 = 60
    int caso3[] = {65, 80};         // idosos → 2*15 = 30
    int caso4[] = {10, 25, 70};     // mistura → 10 + 30 + 15 = 55

    //CASO 1
    mu_check(30==calculaIngressos(caso1, 3));
    //CASO 2
    mu_check(60==calculaIngressos(caso2, 2));
    //CASO 3
    mu_check(30==calculaIngressos(caso3, 2));
    //CASO 4
    mu_check(55==calculaIngressos(caso4, 3));
}
```

Acima estamos encapsulando o teste da função “calculaIngressos()”, criando quatro casos para testar as validades retorno de “calculaIngressos()”, através da função “mu_check()” que retorna a validade da condição estabelecida.

```c
MU_TEST(test_entrada){
    criaArquivoEntrada(4);
    mu_check(4 ==entrada_teste(4));
    mu_check(-2 ==entrada_teste(7));
    mu_check(2 ==entrada_teste(2));
}
```

Acima estamos testando a entrada do usuário, criando um arquivo “entrada.txt” e escrevendo o parâmetro passado pela chamada da função conforme a função abaixo:

```c
void criaArquivoEntrada(int tam){
    FILE *file;
    file = fopen("entrada.txt","w");
    fprintf(file,"%d\n",tam);
    fclose(file);
}
```

Depois de criado o arquivo nos vamos ler o arquivo e ele tem que ser igual ao conteúdo criado no arquivo “entrada.txt”, conforme a função abaixo:

```c
int entrada_teste(){
    int qtdBi;
    FILE *file;
    file = fopen("entrada.txt","r");

    if (!file) {
        perror("Erro ao abrir arquivo");
        return -1;
    }

    if (fscanf(file, "%d", &qtdBi) != 1) {
        fclose(file);
        printf("\nErro ao ler número do arquivo.\n");
        return -1;
    }
    fclose(file);
    if(qtdBi>5){
        printf("\n%d",qtdBi);
        printf("\nQuantidade superior a 5, que é quantidade máxima permitida.\n");
        return -2;
    }
    return qtdBi;
}
```

```c
MU_TEST(test_entradaIngressos){
    int caso1[] = {5, 10, 12};
    int caso2[] = {20, 30};

    criaArquivoEntradaIngressos(3,caso1);
    mu_check(1 == entradaIngressos_teste(3));
    criaArquivoEntradaIngressos(2,caso2);
    mu_check(1 == entradaIngressos_teste(3));

}
```

Acima estamos encapsulando a função “entradaIngressos_teste()”, que simula a execução da função “entradaIngressos()” conforme os códigos abaixo.

```c
int *entradaIngressos(int ingressos[],int tam){
    setlocale(LC_ALL, "Portuguese");
    for(int i=0;i<tam;i++){
        do{
        printf("\nInforme a idade do ingresso Nº %d: ",i+1);
        scanf("%d", &ingressos[i]);
        }while(ingressos[i]>100 || ingressos[i]<0);
    }
    return ingressos;
}
```

```c
int entradaIngressos_teste(int tam){
    FILE *file;
    file = fopen("idades.txt","r");
    int valor[tam];

    for(int i=0;i<tam;i++){
        if (fscanf(file, "%d", &valor[i]) == 1) {
        }else{
            printf("\nErro ao ler o número [%d]",i);
            return 0; // falso
        }
    }
    fclose(file);
    return 1; //verdadeiro
}
```

Depois da criação do arquivo “idade.txt” através da função “criaArquivoEntradaIngressos()”, através da função “entradaIngresso_teste()” e possível ler o arquivo criado anteriormente e retornando a validade da operação.

```c
MU_TEST_SUITE(testaGeral){

    MU_RUN_TEST(test_calculaIngressos);
    MU_RUN_TEST(test_entrada);
    MU_RUN_TEST(test_entradaIngressos);
}
```

Acima encontrasse encapsulada a execução dos três testes.

```c
int main() {

    MU_RUN_SUITE(testaGeral);
    MU_REPORT();
    setlocale(LC_ALL, "Portuguese");
    int qtdBi = entrada();
    int *ingressos = malloc(qtdBi*sizeof(int));
    ingressos=entradaIngressos(ingressos,qtdBi);
    int soma = calculaIngressos(ingressos,qtdBi);
    if(soma==30){
        printf("O total do ingresso ficou em R$%d,00",soma);
    }
    else{
        printf("O total dos ingressos ficou em R$%d,00",soma);
    }
   return MU_EXIT_CODE;
}
```

Acima encontrasse a função “main()” onde de inicio e executado todos os teste através da função “MU_RUN_SUITE(testaGeral)” e ao final da função “main()” vai retornar o relatório.

Abaixo se encontra a evidência do teste, em tempo de execução: