|  |  |  |
| --- | --- | --- |
| **Departamento de Matemática da Universidade de Coimbra** | | |
| **2021 - 2022** | **Programação Orientada para os Objetos** | **Projeto 1** |

**Relatório do Projeto 1**

**Classe IGD**

**Trabalho realizado por:** Margarida Biscaia e André Couto, Grupo 7.

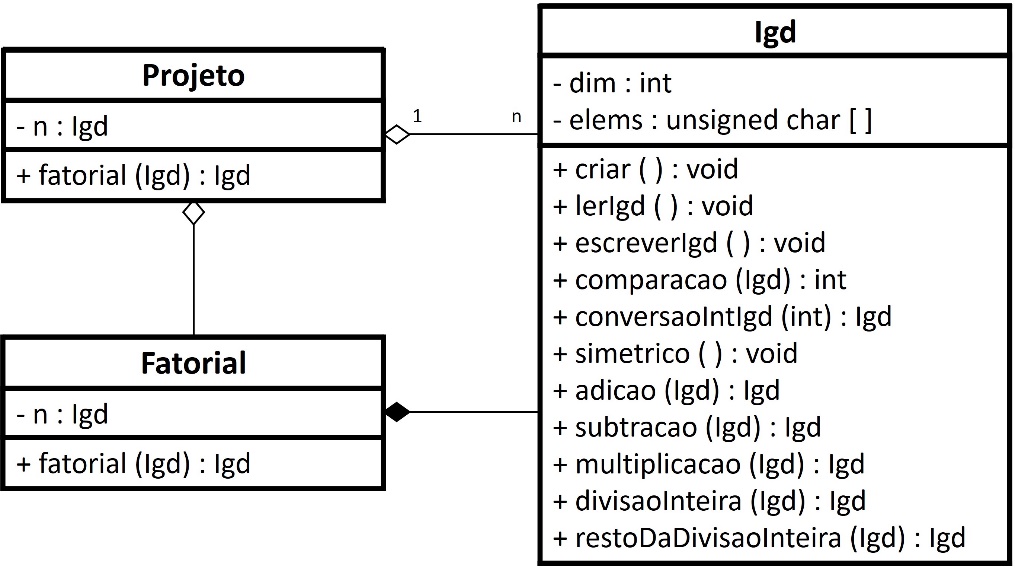
**Especificação:** Dado um n inteiro, construir um programa que receba n, crie um objeto igd, instância da classe IGD, e calcule e escreva o valor de n! .

**entradas:** n inteiro

**saídas:** fatorial de n

**Utilização:** A interface tem duas partes. Primeiro, o utilizador escreve dois inteiros e são apresentadas as operações elementares. Depois, escreve outro inteiro e é lhe devolvido o seu fatorial.

**Diagrama UML:**

****

**Algoritmo:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | criar(n)  lerIgd(n)  criar(m)  lerIgd(m)  adicao(n,m)  subtracao(n,m)  multiplicacao(n,m)  divisaoInteria(n,m)  restoDaDivisaoInteira(n,m)  simétrico(n)  criar(igd)  lerIgd(igd)  fatorial(igd) |

**- Class Igd**

**. Private:** Definimos a dimensão do vetor, dim, inteira e o vetor dos elementos, \*elems, de unsigned char.

**. Public:**

Definimos um construtor que constrói um Igd vazio e um construtor por cópia.

Definimos um destrutor, que elimina o espaço alocado para o vetor elems.

A função *criar*, inicializa a dimensão a 0, aloca memória para um vetor de elems, com espaço para MAXN elementos (MAXN é uma constante previamente definida) e inicializa os elementos do vetor a 0.

A função *lerIgd* pede ao utilizador um inteiro, que é gravado como string e posteriormente copiado para um Igd.

A função *escreverIgd* escreve todos os elementos do vetor elems, tendo em conta se o Igd é ou não negativo.

A função *comparacao* vai comparar um Igd v2 com um Igd v1, e devolver 1, 0 ou -1. Definimos que devolve 0 se v1=v2, 1 se v1>v2 e -1 se v1<v2. Temos assim vários casos. Se a dimensão de v1 for maior que a dimensão de v2, devolve logo 1. Se a dimensão de v1 for menor do que a dimensão de v2, devolve logo -1. Se as dimensões forem iguais, vamos analisar elemento a elemento. Se encontrarmos um elemento v1[i] > v2[i], devolve 1, se encontrarmos v1[i] < v2[i], devolve -1. Se percorremos os vetores todos significa que todos os elementos são iguais e assim devolve 0.

A função *conversaoIntIgd* vai converter um inteiro m num Igd, copiando inicialmente m para uma string e de seguida copiando essa string para um Igd.

A função *simetrico* analisa dois casos. Se o primeiro elemento de elems for o carater ‘-‘, então o número é negativo e retiramos o sinal recuando todos os elementos uma posição. Caso contrário avançamos todos os elementos uma posição e o primeiro elemento passa a ser ‘-‘.

A função *adicao* é uma implementação do algoritmo da adição com transportes, adaptado para Igd. Começamos por colocar os Igd’s com a mesma dimensão, “alinhando” assim os algarismos das unidades, dezenas… (exemplo: para somar [5] com [2 5 0] fazemos [5] -> [0 0 5]) Fazemos isto em dois passos. Copiamos os valores iniciais para o fim do novo vetor e de seguida colocamos os restantes valores iniciais a 0. Podemos agora somar componente a componente. Se a soma for superior a 9, então vamos implementar o mecanismo dos transportes, isto é, dividimos a soma no seu algarismo das unidades e no das dezenas, sendo que o das dezenas será transportado para a soma das componentes seguintes. No fim, se o primeiro elemento do resultado for superior a 9 simplificamo-lo em unidades e dezenas e avançamos o vetor uma posição.

A função *subtracao* é uma implementação do algoritmo da subtração com transportes, adaptado para Igd. Caso v1=v2, então o resultado é 0. Caso contrário, começamos por colocar os Igd’s com a mesma dimensão, “alinhando” assim os algarismos das unidades, dezenas, …, tal como na função *adicao*. É necessário separar a subtração em dois casos, para v1>v2 e v1<v2. A implementação é semelhante. Vamos então subtrair componente a componente utilizando o mecanismo dos transportes. No caso da subtração, os transportes serão incrementados ao valor da componente do menor vetor. Por fim, se o resultado for do tipo [0 0 5] vamos reescrever o vetor com [5]. Caso v1>v2, o resultado será negativo.

A função *multiplicacao* é uma implementação do algoritmo da multiplicação com transportes, adaptado para Igd. Vamos então fixando v2.elems[j], j=v2.dim e j--, multiplicando-o sucessivamente por cada elemento de v1, começando pelo último. Em cada iteração criamos um Igd auxiliar igual ao resultado inicial. No fim da iteração, vamos somar esse auxiliar ao novo resultado obtido. Para começar uma nova iteração, acrescentamos um 0 a v1 e diminuímos a dimensão de v2. (Isto porque para fazer [2 5 0]\*[5 6] fazemos [2 5 0]\*[6] e de seguida [2 5 0 0]\*[5], somando os resultados obtidos).

A função *divisaoInteira* é dependente da *multiplicacao*. Se v1<v2 devolve o Igd 0. Caso contrário, enquanto quociente\*v2 < v1, incrementamos o quociente. No fim, se tivermos quociente\*v2 = v1, o quociente obtido é o resultado. Caso contrário, o resultado é quociente-1.

A função *restoDaDivisaoInteira* devolve v1-*divisaoInteira*(entre v1 e v2)\*v2.

**- Função fatorial**

A função *fatorial* recebe um Igd n e devolve o respetivo fatorial. Implementámos a versão recursiva da função fatorial adaptada para Igd. Se n for 0 o fatorial devolve 1, caso contrário fazemos n\*fatorial(n-1).

**Particularidades:** O programa não aceita inteiros negativos, pois não existe fatorial de um número negativo.

**Exemplo:**

**Uma imagem com texto

Descrição gerada automaticamente**