CONCURSO P2

Soma de sequências (Concurso PI_2223_P2 no Mooshak)

O tema deste concurso é **soma de sequências**, mas é um trabalho de programação, não de aritmética. O que nos interessa é, continuar o desenvolvimento ganhando prática na programação de funções, em C. Cada nova função neste guião relaciona-se com as anteriores e ao programar essa nova função, devemos sempre tirar partido dessa circunstância. Por outras palavras, não queremos, para cada nova função, descobrir e programar a fórmula matemática que facilmente encontraremos, se a procurarmos. Queremos é, percebendo o que cada nova função deve calcular, exprimir esse cálculo usando as funções anteriores, sempre que possível.

Programa A – Soma dos N primeiros números inteiros positivos

Nas aulas teóricas, estudámos o problema da soma de números inteiros consecutivos. Hoje, começamos por um caso particular desse problema: somar os **N** primeiros números inteiros positivos: 1 + 2 + 3 + … + (**N**-1) + **N**. No entanto, agora, como exercício, não queremos usar as fórmulas da matemática, mas sim as "fórmulas" da programação. Em programação pensamos assim: "quero calcular a soma dos primeiros **N** números inteiros positivos? Então, primeiro vou somar os **N-1** primeiros números inteiros positivos e, depois, adiciono o número **N** à soma dos **N-1** primeiros números positivos; ah, e se N for zero não preciso fazer nada: a soma dos zero primeiros números inteiros positivos é zero". Pois bem: programe uma função **sum_positive_integers**, que implemente esta maneira de realizar os cálculos. A função terá um argumento, que representa o número de números a somar. Programe também uma função de teste iterativa, que em cada passo aceita um valor para N e escreve a soma dos N primeiros inteiros positivos, recorrendo à função **sum_positive_integers**. Submeta no **problema A do concurso PI_2122_P2**.

Nota: nos problemas deste guião, as funções de teste são iterativas. É essa a norma, a partir de agora.

Programa B – Média dos N primeiros números inteiros positivos

O segundo exercício é do mesmo género. Queremos agora escrever uma função **avg_positive_integers** que calcula a media da soma dos N primeiros números inteiros. Aqui a

média será o valor real, pelo que a função deverá devolver um valor **double** e terá apenas um argumento, que representa o número de números para efetuar a média.

Programe esta função **avg_positive_integers** recorrendo ao programado anteriormente na função **sum_positive_integers**, criando uma nova função que permita devolver valores **double,** com o nome **sum_positive_integers_double**. Programe ainda uma função de teste iterativa, que aceite o valor dos **N** primeiros números e que execute a função **avg_positive_integers** imprimindo como resultado um valor real (**double**)**.**

Submeta no **Problema B do concurso PI_2223_P2**.

Nota: nos exercícios deste guião, usamos sempre o especificador de conversão **%f** para escrever números **double**.

Programa C - Soma dos múltiplos

O terceiro exercício é também do mesmo género. Queremos agora escrever uma função **sum_multiples** que calcula a soma dos **N** primeiros múltiplos de um dado número natural, **R**. A função terá dois argumentos, **r** para o número cujos múltiplos queremos somar e **n** para o número de parcelas.

Programe esta função **sum_multiples**, recorrendo de novo à função **sum_positive_integers**, e programe também uma função de teste iterativa, que em cada passo aceita novos valores para **r** e **n**, por esta ordem.

Submeta no **problema C**.

Nota: o primeiro múltiplo de qualquer número é zero; o segundo é o próprio número; o terceiro é o número vezes 2; etc.

Sugestão: no problema A anterior, programámos a soma dos **N** primeiros números inteiros positivos. Aplicando a mesma técnica, programe uma função **sum_naturals** para somar os N primeiros números naturais, 0 + 1 + 2 + 3 + … + (**N**-1), reparando que o último dos **N** primeiros números naturais é **N**-1. Esta função, **sum_naturals** será útil no presente exercício sobre a soma dos múltiplos e também em alguns dos exercícios seguintes.

Programa D – Progressões aritméticas

Tanto a sequência dos **N** primeiros números inteiros positivos, como a soma dos **N** primeiros números inteiros positivos, como a sequência dos números inteiros entre **X** e **Y,** como ainda a sequência dos **N** primeiros múltiplos de **R**, são progressões aritméticas. De facto, uma

progressão aritmética é uma sequência de números em que a diferença entre dois elementos consecutivos é sempre a mesma.

Em geral, uma progressão aritmética finita fica determinada pelo valor do seu primeiro elemento, pela tal diferença constante, que é chamada razão, e pelo número de elementos. Pois bem: programe uma função **sum_progression** para calcular a soma dos elementos de uma progressão aritmética. A função terá três argumentos: **x0**, representando o primeiro elemento da progressão; **r**, representando a razão, e **n**, representando o número de elementos.

Programe também a função de teste iterativa, que agora aceita três valores em cada passo do ciclo: **x0**, **r** e **n**, por esta ordem.

Submeta no **problema D**.

Note bem: a ideia aqui não é usar a fórmula matemática. É sim, pensar funcionalmente e relacionar esta função com as anteriores.

Sugestão: que sucessão obtemos se subtrairmos a cada elemento da progressão aritmética o valor do primeiro elemento?

Problema E – Programação alternativa

Há uma outra maneira interessante de programar a soma da progressão aritmética, recorrendo a um argumento análogo ao que usámos para a função **sum_positive_integers**.

Vejamos: queremos somar os **N** primeiros elementos da progressão aritmética cujo primeiro elemento é **X0** e cuja razão é **R**. Admitindo que **N** > 0, então, se apagarmos o primeiro elemento dessa progressão aritmética ficamos com uma progressão aritmética com **N**-1 elementos, com a mesma razão e cujo primeiro elemento é **X0+R**.

Esta observação conduz-nos diretamente à seguinte função C, agora programada para números **double**, para maior generalidade:

```
double sum_progression_dbl(double x0, double r, int n)
{
  return n == 0 ? 0 : x0 + sum_progression_dbl(x0 + r, r, n-1);
}
```

Repare bem: a expressão **sum_progression_dbl(x0 + r, r, n-1)** calcula a soma da progressão aritmética cujo primeiro termo é **x0+r**, cuja razão é r e cujo número de elementos é **n-1**. Escreva uma função de teste para a função **sum_progression_dbl**, sem esquecer que agora os dois primeiros argumentos, e também o resultado, são **double**. O terceiro argumento, que representa o comprimento da sequência, é um número inteiro. Em cada linha do input vêm

valores para **x0**, **r** e **n**, por esta ordem. Ao escrever o resultado, use o especificador de conversão **%f**.

Submeta no **problema E**.

Nota: nos exercícios deste guião, usamos sempre o especificador de conversão **%f** para escrever números **double**.

Problema F – Soma dos quadrados

Usando a técnica da função **sum_progression_dbl**, programe uma função **sum_squares_from** para calcular a soma $x^2 + (x+1)^2 + (x+2)^2 + \ldots + (x+n-1)^2$, para **x** e **n** dados. Admita, de novo, que se trata de uma sequência de números **double**.

Programe uma função de teste, análoga às anteriores, para a função **sum_squares_from**.

Submeta no **problema F**.

Problema G – Soma das potências

Queremos agora uma função **sum_powers_from** análoga à função **sum_squares_from**, que, dados **x** e **n** como antes e ainda um expoente **y**, calcula a soma **x**y + (x+1)**y + … + (x+n-1)**y**. (Aqui, a expressão **x**y** significa **x** elevado a **y**.

Note que o operador ****** não existe em C. Em C, para potências arbitrárias, recorremos à função de biblioteca **pow**. Por exemplo, para calcular **x**y**, usamos a expressão **pow(x, y)** .

Programe uma função de teste análoga às anteriores. Em cada linha do input vêm os valores de **x**, **y** e **n**, por esta ordem.

Submeta no **problema F**.

Nota: quando usar a função **pow**, insira a diretiva **#include <math.h>**.

Problema H – Soma dos inversos

Programe uma função **sum_inverses**, que calcula as soma dos inversos dos **N** primeiros números inteiros positivos, **1 + 1/2 + 1/3 + … + 1/N**, recorrendo diretamente à função **sum_powers_from**.

Programe uma função de teste análoga às anteriores.

Submeta no **problema H**.

Note bem: a sua solução não deve reproduzir o argumento usado para programar a função **sum_powers_from**. Deve sim invocar esta função, fixando o valor de alguns dos argumentos.

Problema I – Soma dos inversos dos quadrados

Programe uma função **sum_inverse_squares**, que calcula a soma dos inversos dos quadrados dos **N** primeiros números inteiros positivos, **$1 + 1/4 + 1/9 + \ldots + 1/N^2$**.

Programe uma função de teste análoga às anteriores, e submeta no problema I do concurso PI_2122_P2.

Note bem: aplica-se aqui também a observação feita no problema anterior. ~

Sequências importantes para a análise de programas

Algumas das sequências que estudámos hoje têm muito interesse em programação, sobretudo para estudar a complexidade dos programas, isto é, o número de operações elementares que os programas realizam para fazer os cálculos:

- Soma triangular: $1+2+3+4+\ldots+N = (N\ (N+1))\ /\ 2 \sim N^2/\ 2$
- Soma harmónica: $1+1/2+1/3+1/4+\ldots+1/N \sim \ln N$

O til representa a ideia de "aproximadamente igual". A função ln é o logaritmo natural. É representada em C pela função de biblioteca **log**.

De facto, a soma dos N primeiros números inteiros positivos é (N (N+1)) / 2, que é aproximadamente igual a $N^2/\ 2$. Do mesmo modo, a soma dos inversos do N primeiros números inteiros positivos é aproximadamente igual a logaritmo natural de N. Uma outra sequência cuja soma é interessante, mas que não surgiu ainda, é a progressão geométrica: x**0, x**1, x**2, …, , x**(n-1). Quando x vale 2 temos 1, 2, 4, 8, … 2**(N-1):

- Soma geométrica: 1+2+4+8+…+2**N = 2**N - 1 ~ 2**N 6