Somas de sequências

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. No entanto, agora, como exercício, não queremos usar as fórmulas da matemática, mas sim as "fórmulas" da programação. Em programação pensamos assim: "queres a soma dos primeiros **N** números inteiros positivos? Então, primeiro soma os **N-1** primeiros números inteiros positivos e, depois, adiciona **N** à soma dos **N-1** primeiros números positivos; ah, e se **N** for zero não faças 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 P2_1819.

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

Soma dos múltiplos

O segundo exercício de hoje é 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, 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 *B* do concurso P2_1819.

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

Progressões aritméticas

Tanto a sequência 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 *C* do concurso P2_1819.

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 o valor do primeiro elemento?

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);
}
```

Escreva uma função de teste para a função `sum_arithmetic`, 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 formato `%f`. Submeta no problema *D* do concurso P2_1819.
Nota: nestes exercícios, usamos sempre o especificador `%f` para escrever números `double`.

Soma dos quadrados

Usando a técnica da função `sum_arithmetic`, programe uma função `sum_squares_from` para calcular a soma $x^2 + (x+1)^2 + \ldots + (x+n-1)^2$, para `x` e `n` dados. Neste caso, admita também 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 *E* do concurso P2_1819.

Nota: esta função é diferente da função do quarto problema do guião anterior, que calculava a soma dos quadrados dos números inteiros entre dois números dados. Aqui, os dados são o primeiro número inteiro da sequência e o número de números (e não o último número inteiro da sequência).

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 $x^y + (x+1)^y + \ldots + (x+n-1)^y$. Para a potência, recorremos à função de biblioteca `pow`.
Programe uma função de teste análoga às anteriores e submeta no problema *F* do concurso P2_1819. Em cada linha do input vêm os valores de `x`, `y` e `n`, por esta ordem.
Nota: para usar a função `pow`, insira a diretiva `#include <math.h>`.

Soma dos inversos

Programe uma função `test_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, e submeta no problema *G* do concurso P2_1819.

Soma dos inversos dos quadrados

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

Programe uma função de teste análoga às anteriores, e submeta no problema *H* do concurso P2_1819.

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 eles realizam. Eis três delas, para referência:

- Soma triangular: $1+2+3+4+\ldots+N = N * (N+1) / 2 \sim N^2 / 2$
- Soma harmónica: $1+1/2+1/3+1/4+\ldots+1/N \sim \mathit{ln}\ N$
- Soma geométrica: $1+2+4+8+\ldots+2^N = 2^{N+1} - 1 \sim 2^{N+1}$
  Nota: a função *ln* é o logaritmo natural. É representada em C pela função de biblioteca `log`.