Aula 2 · Condições, Repetições, Operações Lógicas e de Relação

Introdução à Programação de Maratona

Fernando Kiotheka

UFPR

03/05/2020

Problema: Conta (URI 1866)

Dois amigos pedem ao atendente de uma lanchonete propor um desafio, de modo que quem acertasse mais, não precisaria pagar a conta. Então foi proposto o seguinte: Dado o seguinte somatório abaixo, informa o resultado, com uma quantidade de termos no mesmo:

$$S = 1 - 1 + 1 - 1 + 1 - 1 + 1 - 1 + \ldots$$

Escreva um programa que, dada uma quantidade de termos, informar o resultado do somatório acima.

Problema: Conta (URI 1866) continuado

Entrada

Um número inteiro C ($1 \leq C \leq 10^5$) será informado, que será a quantidade de casos de teste. Cada caso de teste inicia com um número inteiro N ($1 \leq N \leq 1000$), indicando a quantidade de termos da soma.

Saída

Para cada caso de teste imprima um número S, que é o resultado da soma dos N termos da expressão.

Problema: Conta (URI 1866) continuado

Exemplos de Entrada	Exemplos de Saída
conta-1.in	conta-1.out
3 11 7 18	1 1 0

Casos de teste

Um padrão que você vai encontrar muitas vezes é esse conceito de *casos de teste*. É preciso prestar atenção porque todo o cálculo que você já fez precisa estar limpo para cada novo caso de teste.
Eles são apresentados, geralmente, em uma de duas formas:

- Número de casos é informado e então são apresentados
- Um ou mais valores "sentinela" indicam o final da entrada

Neste problema temos o primeiro, então vamos ver o que fazer.

Repetindo, repetindo, repetindo, repetindo...

Apresentamos uma das construções da programação estruturada:

Código 1 Comando de repetição `while`, ou laço `while`

```
while (condicao) {
    // código
}
```

while traduz para **enquanto**, e funciona da seguinte forma:

1. Verifica se a condição é falsa, se for, vá para o passo 4
2. Executa o código
3. Volta para o passo 1
4. Sai da repetição

Condição, é?

Faltou explicar um tipo de dados na aula passada: o `bool`. Seu nome vem de George Boole, que inventou a álgebra booleana que você vai dominar em Circuitos Lógicos.

Tá mas, qual a graça?

Nenhuma. um booleano é só um tipo que representa dois valores:

- Verdadeiro (`true`, `1`)
- Falso (`false`, `0`)

Esse é o tipo de dados da condição. Mas daonde a gente tira isso?

Operações de relação ou comparação

Operação	Símbolo	Exemplo
Menor que	`<`	`3 < 4, x < 1`
Menor ou igual a	`<=`	`3 <= 4, y <= x`
Igual a	`==`	`3 == 4, y == x`
Diferente de	`!=`	`3 != 4, y != x`
Maior que	`>`	`3 > 4, y > x`
Maior ou igual a	`>=`	`3 >= 2, x >= 2`

Com esses operadores, podemos fazer perguntas às nossas variáveis, e determinar se a gente precisa repetir ou não.

Então vamos repetir!

Usando das variáveis, podemos criar a seguinte estrutura para executar nosso código C vezes, uma vez para cada caso de teste:

Código 2 `while-basico.cpp`

```
int C;
cin >> C;

while (C > 0) {
    // código para cada caso de teste
    C = C - 1;
}
```

Com isso e com o que já vimos, resolvemos o problema. Tente!

Solução da problema da Conta (URI 1866)

Código 3 `conta-solucao.cpp`

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int C;
    cin >> C;

    while (C > 0) {
        int N;
        cin >> N;
        cout << N % 2 << "\n";
        C = C - 1;
    }
}
```

Problema: Tri-du (URI 1933)

Tri-du é um jogo de cartas derivado do popular jogo de Truco. O jogo utiliza um baralho normal de 52 cartas, com treze cartas de cada naipe, mas os naipes são ignorados. Apenas o valor das cartas, considerados como inteiros de 1 a 13, são utilizados.
No jogo, cada jogador recebe três cartas. As regras são simples:

- Um trio (três cartas de mesmo valor) ganha de uma dupla (duas cartas de mesmo valor).
- Um trio formado por cartas de maior valor ganha de um trio formado por cartas de menor valor.
- Uma dupla formada por cartas de maior valor ganha de uma dupla formada por cartas de menor valor.

Problema: Tri-du (URI 1933) continuado

Note que o jogo pode não ter ganhador em muitas situações; nesses casos, as cartas distribuídas são devolvidas ao baralho, que é embaralhado e uma nova partida é iniciada
Um jogador já recebeu duas das cartas que deve receber, e conhece seus valores. Sua tarefa é escrever um programa para determinar qual o valor da terceira carta que maximiza a probabilidade de esse jogador ganhar o jogo.

Problema: Tri-du (URI 1933) continuado

Entrada

A entrada consiste de uma única linha que contém dois inteiros, A ($1 \leq A \leq 13$) e B ($1 \leq B \leq 13$) indicando os valores das duas primeiras cartas recebidas.

Saída

Seu programa deve produzir uma única linha com um inteiro representando o valor da carta que maximiza a probabilidade de o jogador ganhar a partida.

Problema: Tri-du (URI 1933) continuado

Exemplos de Entrada	Exemplos de Saída
`tridu-1.in`	`tridu-1.out`
`10 7`	`10`
`tridu-2.in`	`tridu-2.out`
`2 2`	`2`

Hmmm... que?

Esse problema parece complicado a primeira vista.
Ele é o problema fácil da prova da XX Maratona de Programação da SBC 2015, e nessas provas sempre tem um problema parecido com esse: Fácil e com enunciado complicado.
Vamos resolver então olhando para as regras do enunciado:

- Um trio (três cartas de mesmo valor) ganha de uma dupla (duas cartas de mesmo valor).
- Um trio formado por cartas de maior valor ganha de um trio formado por cartas de menor valor.
- Uma dupla formada por cartas de maior valor ganha de uma dupla formada por cartas de menor valor.

O que devemos fazer?

Analisando as regas do jogo

Precisamos então ter uma série de intuições:

1. Para ganhar, o ideal é ter três cartas iguais, mas controlamos apenas uma, então se forem iguais, escolhemos uma igual
2. Se não tivermos duas iguais, precisamos escolher uma das duas para que possamos ganhar por meio da regra da dupla
3. Se ambos usarem duplas, a maior dupla vence

Então está claro o que precisamos fazer: Pegar a maior das duas cartas, o máximo das duas cartas. Mas como que a gente escolhe?

Desviando, desviando, desviando, ...

Para escolher, nós utilizamos outra construção, o `if`:

Código 4 Comando de desvio condicional `if`

```
if (condicao) {
    // código se verdadeiro
} else {
    // código se falso
}
```

if que se traduz **se**, funciona da seguinte maneira:

1. Verifica se condição é falsa, se for, vá para o passo 4
2. Executa o código do bloco verdadeiro
3. Vá para o passo 5
4. Executa o código do bloco falso
5. Sai da condição

Opcionalidade do `else`

O parte do `else` é opcional, você pode fazer só isso se quiser:

```
if (condicao) {
    // código se verdadeiro
}
```

Agora use esse conhecimento sobre o `if` e resolva o problema Tri-du!

Solução do problema Tri-du

Código 5 `tridu-solucao.cpp`

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int N;
    int M;
    cin >> N >> M;
    if (N > M) {
        cout << N << "\n";
    } else {
        cout << M << "\n";
    }
}
```

Solução alternativa do problema Tri-du

Código 6 `tridu-solucao-alt.cpp`

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int N;
    int M;
    cin >> N >> M;
    if (M > N) {
        N = M;
    }
    cout << N << "\n";
}
```

Funções auxiliares

Para resolver esse e muitos outros problemas, podemos usar funções auxiliares disponíveis na biblioteca padrão:

Função	Nome	Exemplo
Máximo	`max`	`max(3, 4)`, `max(3, 4, 5)`
Mínimo	`min`	`min(3, 4)`, `min(1, 2, 3)`
Absoluto	`abs`	`abs(1, 2)`, `abs(y - x)`
Maior Divisor Comum	`__gcd`	`__gcd(3, 4)`, `__gcd(x, y)`

Podendo ser usadas para escrever código de maneira mais simples.

Solução do problema Tri-du com `max`

Código 7 `tridu-solucao-max.cpp`

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int N;
    int M;
    cin >> N >> M;
    cout << max(N, M) << "\n";
}
```

Problema: Notas da Prova (URI 2344)

Rosy é uma talentosa professora do Ensino Médio que já ganhou muitos prêmios pela qualidade de sua aula. Seu reconhecimento foi tamanho que foi convidada a dar aulas em uma escola da Inglaterra. Mesmo falando bem inglês, Rosy ficou um pouco apreensiva com a responsabilidade, mas resolveu aceitar a proposta e encará-la como um bom desafio.
Tudo ocorreu bem para Rosy até o dia da prova. Acostumada a dar notas de 0 (zero) a 100 (cem), ela fez o mesmo na primeira prova dos alunos da Inglaterra. No entanto, os alunos acharam estranho, pois na Inglaterra o sistema de notas é diferente: as notas devem ser dadas como conceitos de A a E. O conceito A é o mais alto, enquanto o conceito E é o mais baixo.

Problema: Notas da Prova (URI 2344) continuado

Conversando com outros professores, ela recebeu a sugestão de utilizar a seguinte tabela, relacionando as notas numéricas com as notas de conceitos:

Nota	Conceito
0	E
1 a 35	D
36 a 60	C
61 a 85	B
86 a 100	A

O problema é que Rosy já deu as notas no sistema numérico, e terá que converter as notas para o sistema de letras. Porém, Rosy precisa preparar as próximas aulas (para manter a qualidade que a tornou reconhecida), e não tem tempo suficiente para fazer a conversão das notas manualmente.
Você deve escrever um programa que recebe uma nota no sistema numérico e determina o conceito correspondente.

Problema: Notas da Prova (URI 2344) continuado

Entrada

A entrada contém um único conjunto de testes, que deve ser lido do dispositivo de entrada padrão (normalmente o teclado). A entrada contém uma única linha com um número inteiro N ($0 \leq N \leq 100$), representando uma nota de prova no sistema numérico.

Saída

Seu programa deve imprimir, na saída padrão, uma letra (A, B, C, D, ou E em maiúsculas) representando o conceito correspondente à nota dada na entrada.

Problema: Notas da Prova (URI 2344) continuado

Exemplos de Entrada	Exemplos de Saída
notas-prova-1.in	notas-prova-1.out
12	D
notas-prova-2.in	notas-prova-2.out
87	A
notas-prova-3.in	notas-prova-3.out
0	E

Parece fácil, mas teremos que aninhar condições!

Uma das coisas que você vai precisar fazer para resolver esse problema é colocar um `if` dentro do outro:

```
if (condicao1) {
    // se condicao1
} else {
    if (condicao2) {
        // se condicao1 falsa e condicao2 verdadeira
    } else {
        // se condicao1 falsa e condicao2 falsa
    }
}
```

Se você indentar certo (e você deve!), o seu código vai começar a ficar muito pra direita... Além de que se acumularão muitos }!

Uma forma um pouco melhor de ler

Os criadores do C e do C++ deixaram opcional o uso de {s e }s para `ifs`, `whiles` e outros comandos com um comando só dentro:

```
if (condicao)
    x = 1;
```

Como é fácil esquecer que isso só se aplica a um comando, recomendo sempre colocar {s e }s, mas esse caso é clara exceção:

Código 8 `ifs-aninhados.cpp`

```
if (condicao1) {
    // se condicao1
} else if (condicao2) {
    // se condicao1 falsa e condicao2 verdadeira
} else {
    // se condicao1 falsa e condicao2 falsa
}
```

Operações lógicas

Operações lógicas são operações feitas apenas com booleanos. Você as verá mais a fundo em Circuitos Lógicos, mas sendo essenciais, precisamos delas para resolver nosso problema:

Operação	Símbolo	Exemplo
Negação	`!`	`!cond`, `!(x > 5)`
E	`&&`	`true && false`, `x < 0 && x > 2`
Ou	`||`	`x > 0 || x < 2`, `true || x > 4`

Mas como cada uma funciona?

Operação lógica de negação

A operação lógica de negação `!` inverte o valor booleano. Podemos representá-la usando uma tabela verdade (`0` é `false`, e `1` é `true`):

Entrada	Saída
0	1
1	0

Podemos também, implementá-la usando um `if`:

```
bool entrada, saida;

if (entrada) {
  saida = false;
} else {
  saida = true;
}
```

Operação lógica e

A operação lógica && é verdadeira se ambos valores forem também.

Entrada 1	Entrada 2	Saída
0	0	0
0	1	0
1	0	0
1	1	1

Ela é usada da mesma forma que a gente usa a conjunção “e”.

Uso implícito do e

As vezes usamos o "e" implicitamente, por exemplo em:

$$0 \leq x \leq 10$$

Neste caso, devemos sempre escrever:

```
(0 <= x && x <= 10)
```

E nunca:

```
(0 <= x <= 10)
```

Que surpreendemente compila, mas não faz o que você quer quando você usa C++ (em Python funciona, por exemplo).

Implementação usando `if` da operação e

Podemos também implementar usando um `if`, sem problemas:

```
bool entrada1, entrada2, saida;

saida = false;
if (entrada1) {
    if (entrada2) {
        saida = true;
    }
}
```

E o código para resolver esse problema sem o && provavelmente teria isso em algum lugar.

Operação lógica ou

A operação lógica || é verdadeira se qualquer valor for também.

Entrada 1	Entrada 2	Saída
0	0	0
0	1	1
1	0	1
1	1	1

Cuidado porque esse geralmente não é o uso da conjunção "ou". Este tipo de ou é chamado de "ou inclusivo", porque dita que se os dois valores de entrada são verdadeiros, assim também é a saída.

Implementação usando `if` da operação ou

Também podemos implementar usando `ifs`, se quiser:

```
bool entrada1, entrada2, saida;

if (entrada1) {
    saida = true;
} else {
    if (entrada2) {
        saida = true;
    } else {
        saida = false;
    }
}
```

Mas lembre-se que essas implementações são de caráter didático (já que temos o operador implementado em hardware), mas você pode usá-las para identificar padrões e simplificar o código.

Mas chega de papo!

Já aprendemos mais que o suficiente para resolver esse problema, mas como é um código um pouco pesado de seleção se atente em:

- Testar todos os casos colocados
- Verificar todos os valores para ver se batem com o da tabela
- Não se preocupar com valores fora do intervalo pois eles não serão testados, a entrada é restrita a $0 \leq N \leq 100$.

Mãos a obra!

Solução do problema Notas da Prova (URI 2344)

Código 9 notas-prova-solucao.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int N;
    cin >> N;
    if (N == 0) {
        cout << "E" << "\n";
    } else if (1 <= N && N <= 35) {
        cout << "D" << "\n";
    } else if (36 <= N && N <= 60) {
        cout << "C" << "\n";
    } else if (61 <= N && N <= 85) {
        cout << "B" << "\n";
    } else {
        cout << "A" << "\n";
    }
}
```

Problema: Novo Recorde (URI 2551)

A grande Maratona de Rua de Curitiba irá ocorrer nos próximos dias! Vários atletas estão treinando há dias para o grande dia da corrida. Flávio é um dos atletas que está treinando diariamente para se sair bem na corrida. Ele tem corrido todas as manhãs nas pistas próximas de sua casa.
Os treinos do garoto são monitorados por um aplicativo em seu celular. Após cada treino, Flávio sabe tanto a duração do treino quanto a distância total percorrida. Com essas informações, ele consegue determinar a velocidade média obtida em cada treino. Flávio está muito preocupado com a evolução de seu desempenho nos treinos, e em particular com seu recorde de velocidade média. Tal recorde é batido em um dado treino quando a velocidade média para este treino é maior que todas as velocidades médias obtidas nos treinos anteriores. Ajude Flávio a determinar em quais treinos ele conseguiu bater seu recorde.

Problema: Novo Recorde (URI 2551) continuado

Entrada

A entrada contém vários casos de teste. A primeira linha de cada caso contém um inteiro N ($1 \leq N \leq 30$), o número de treinos feitos. Considere que os treinos foram feitos nos dias $1, 2, \ldots, N$.
As próximas N linhas descrevem os treinos. A linha i ($1 \leq i \leq N$) contém dois inteiros T_i e D_i ($1 \leq T_i, D_i \leq 100$), indicando, respectivamente, a duração do treino (em minutos) e a distância percorrida no treino (em quilômetros).
A entrada termina com fim-de-arquivo (EOF), e o número de casos de teste não ultrapassa 10^3.

Saída

Para cada caso de teste, imprima uma lista de inteiros indicando os dias nos quais o recorde foi batido. Cada dia deve ser impresso em uma linha. Imprima os dias em ordem crescente. Note que o dia 1 sempre deve ser impresso.

Problema: Novo Recorde (URI 2551) continuado

Exemplos de Entrada	Exemplos de Saída
novo-recorde-1.in	novo-recorde-1.out
3 1 1 2 1 2 3 2 2 16 4 20	1 3 1

A entrada termina com fim-de-arquivo?

Esse é um problema que tem o segundo formato de casos de teste, só que o que indica o final da entrada é... o final da entrada! Mas como que a gente obtém isso? Bom, a gente pode fazer assim:

```
int N;
while (cin >> N) {
    // código
}
```

Mas por quê? Bom, devido a propriedades mágicas, o objeto `cin` pode virar um booleano, e ele vira um booleano falso se o fim do arquivo ou algum erro foi encontrado na leitura.
Como `cin >> x >> y >> ...` sempre retorna `cin`, isso funciona magicamente e automaticamente já lê o próximo valor.

Mas como que a gente faz isso no terminal?

No terminal, você ou pode realmente fazer pipe de um arquivo:

```
$ ./a.out <input
$
```

Ou, você pode colocar o conteúdo e depois finalizar com um Control+D, que simboliza o fim da entrada para o processo:

```
$ ./a.out
1 2 3
$
```

Fato curioso: Se você der Control+D sem um processo rodando, é a mesma coisa que fechar o terminal ou dar `exit`, já que é o fim da sua própria entrada para o `bash`.

Um comando novo para preguiçosos: `for`

Você já deve ter visto algum código na internet usando `for`, mas a gente só usou até agora o `while`, isto porque ele é bem mais geral.

Código 10 Comando `for`

```
for (inicializacao; condicao; incremento) {
    // código
}
```

for que traduz para **para** equivale a esse código com `while`:

```
inicializacao;
while (condicao) {
    // código
    incremento;
}
```

Mas então como geralmente se usa o `for`?

Exemplos de `for` para iterar

Todo i em $[0, N-1]$ ou $[0, N[$

```
for (int i = 0; i < N; i++)
```

Todo i em $[1, N]$ ou $[1, N+1[$

```
for (int i = 1; i <= N; i++)
```

Todo i em $[0, N-1]$ ou $[0, N[$ em ordem reversa

```
for (int i = N - 1; i >= 0; i--)
```

Todo j em $[i, N-1]$ ou $[i, N[$

```
for (int j = i; j < N; j++)
```

Todo j tal que $1 \leq i \leq N$, e $i = 2^k$ para algum $k \in \mathbb{Z}$

```
for (int i = 1; i <= N; i *= 2)
```

Todo j tal que $i \leq j < N$, e $j = ik$ para algum $k \in \mathbb{Z}$

```
for (int j = i; j < N; j += i)
```

Vamos a ação!

Usando essas ferramentas já podemos resolver este problema.
Algumas dicas:

- A solução "mais fácil", com números reais é muito perigosa (porque número real no computador é uma mentira)
  - `double` não funciona devido a problemas na otimização
  - `long double` e `float` funcionam neste problema
  - Se for por esse caminho use o mesmo tipo de dados em todas as variáveis para evitar truncamento de inteiro
- Use do poder da álgebra para fazer uma conta que só usa inteiros, daí você vai ter certeza que funciona!
- Preste atenção no intervalo dos índices que precisam ser impressos
- A velocidade média é dada por $v = d/t$, não se esqueça!

Solução do problema Novo Recorde (URI 2551)

Código 11 novo-recorde-solucao.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int main () {
    int N;
    while (cin >> N) {
        int recorde_D = 0, recorde_T = 1;
        for (int i = 1; i <= N; i++) {
            int T, D;
            cin >> T >> D;
            if (D*recorde_T > recorde_D*T) {
                cout << i << "\n";
                recorde_D = D;
                recorde_T = T;
            }
        }
    }
}
```

Solução do problema Novo Recorde (URI 2551) perigosa

Código 12 novo-recorde-solucao-perigosa.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int main () {
    int N;
    while (cin >> N) {
        long double recorde = 0;
        for (int i = 1; i <= N; i++) {
            long double T, D;
            cin >> T >> D;
            long double recorde_atual = D/T;
            if (recorde_atual > recorde) {
                cout << i << "\n";
                recorde = recorde_atual;
            }
        }
    }
}
```

O resto da aula

- Tem mais um problema para resolver, e vocês conseguem resolver apenas com o que vocês já sabem, boa sorte!
- Confiram o site depois para ver uma lista de exercícios para praticar ainda mais!