UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO DE COMPUTADORES

LABORATÓRIO 6

TIPOS INTEIROS

EXERCÍCIOS DE REVISÃO

VOCÊ DEVE ACOMPANHAR PARA OBTER INFORMAÇÕES COMPLEMENTARES

1. Execute e tente entender o que cada instrução do programa abaixo faz.

```
#include <iostream>
using namespace std;

#define black   "\033[7;37;40m"
#define yellow  "\033[1;33;44m"
#define green   "\033[32m"
#define red     "\033[4;31m"
#define foreg   "\033[38;5;154m"
#define backg   "\033[38;5;0;48;5;154m"
#define default "\033[m"

int main()
{
    cout << black  << "Preto no Branco" << default << endl;
    cout << yellow << "Amarelo Intenso com Azul" << default << endl;
    cout << green  << "Verde Normal" << default << endl;
    cout << red    << "Vermelho Sublinhado" << default << endl;
    cout << foreg  << "256 Cores" << default << endl;
    cout << backg  << "256 Cores" << default << endl;

    return 0;
}
```

O programa usa códigos de Escape definidos pela ANSI para mudar as cores do terminal de comandos. Eles são válidos para todos os sistemas operacionais, e definidos por um estilo, cor da letra e cor do fundo:

\033[estilo**;** letra**;** fundo **m**

Estilo: 0 = normal, 1 = bold, 4 = underline, 7 = negative
Letra: 30=black, 31=red, 32=green, 33=yellow, 34=blue, 35=magenta, 36=cyan, 37=white
Fundo: 40=black, 41=red, 42=green, 43=yellow, 44=blue, 45=magenta, 46=cyan, 47=white

Um conjunto estendido (com 256 cores) é suportado por alguns terminais:
Letra: **\033[**38;5;{0-255}**m**
Fundo: **\033[**48;5;{0-255}**m**

EXERCÍCIOS DE FIXAÇÃO

VOCÊ DEVE FAZER OS EXERCÍCIOS PARA FIXAR O CONTEÚDO

1. Corrija o programa abaixo mantendo a constante simbólica ZERO.

```
include  iostream
namespace using std
void main{}
(
   Int total;
   Cout << "Inicialmente total = " << Total << " (ZERO)\n";
   Total = 50;
   cout << "Depois da atribuição total = " << total << endl;
   Return ZERO;
)
```

De forma que a saída seja:

```
Inicialmente total = 0 (ZERO)
Depois da "atribuição" total = 50
```

2. Escreva um programa que peça quantos quilômetros você dirigiu e quantos litros de combustível foram gastos e então mostre o consumo do seu carro em km/litro. É possível obter um resultado sempre correto usando apenas valores e variáveis do tipo inteiro? Faça o teste com o seu programa.

sim, limitando que todos os valores sejam inteiros.

```
Distância percorrida (km): 300
Litros de combustível: 30
O consumo do seu carro foi de 10 km/litro.
```

3. Escreva um programa que crie quatro variáveis, uma para cada um dos tipos inteiros mostrados abaixo. O programa deve mostrar quantos bytes cada tipo ocupa e calcular a quantidade total de memória utilizada pelas variáveis criadas. Use espaços nas cadeias de caracteres para produzir a saída no formato abaixo:

```
short:       2 bytes
int:         4 bytes
long:        4 bytes
long long:   8 bytes
Total:      18 bytes
```

EXERCÍCIOS DE APRENDIZAGEM

VOCÊ DEVE ESCREVER PROGRAMAS PARA REALMENTE APRENDER

1. A empresa Viagens&Turismo possui uma linha de ônibus fazendo o trecho Mossoró-Natal. Considerando que as viagens são feitas sempre dentro de um mesmo dia, elabore um programa que permita ao usuário informar o momento de partida e de chegada (hora e minuto) do ônibus. O programa deve calcular o tempo total da viagem (em horas e minutos).

```
Digite o horário de partida (HH:MM): 10:50
Digite o horário de chegada (HH:MM): 14:20

O tempo total de viagem foi 3 horas e 30 min.
```

2. Utilize a diretiva #define para criar constantes simbólicas para o preço de um pão (R$0,30) e para o preço de um pastel (R$0,25). Na função principal, peça para o usuário informar quantos pães e quantos pasteis ele quer. Utilize as constantes simbólicas que representam os preços dos produtos para fazer o cálculo de quanto o usuário vai pagar, e mostre o resultado na tela.

```
Pães&Cia

Quantos pães? 5
Quantos pastéis? 4

O total das compras é R$2.50
```

3. Modifique o programa abaixo de forma que ele continue usando 16 bits para a variável x e 32 bits para a variável y, mas sem apresentar overflow:

```cpp
#include <iostream>

int main()
{
    short x = 1;
    x = x + 32767;
    std::cout << "x = " << x << std::endl;

    int y = 2'147'483'647;
    y = y + 1;
    std::cout << "y = " << y << std::endl;


    return 0;
}
```

4. Altere o programa para que o resultado seja exibido corretamente tanto no cálculo direto quanto no cálculo feito através da função. Explique o porquê dessa diferença.

```
#include <iostream>
using namespace std;

long long calculo(long long, long long);

int main()
{
	long long resultado = 200530 * 420800;
	cout << "Direto: " << resultado << endl;
	cout << "Função: " << calculo(200530, 420800) << endl;
	return 0;
}

long long calculo(long long a, long long b)
{
	return a * b;
}
```

5. Construa uma função `isShort()` para verificar se um número cabe em 16 bits e uma função `isInt()` para verificar se um número cabe em 32 bits. Para alcançar esse objetivo, as funções devem receber o valor sempre em uma quantidade de bits maior e usar as constantes definidas no arquivo de cabeçalho `<climits>` para verificar se os números estão dentro das faixas MIN e MAX dos respectivos tipos. As funções devem retornar um valor booleano `true` ou `false`. Construa um programa para testar as funções, como nos exemplos de execução abaixo.

Exemplo de execução 1:

```
Digite um valor inteiro: 300
300 cabe em 16 bits
300 cabe em 32 bits
```

Exemplo de execução 2:

```
Digite um valor inteiro: 70000
70000 não cabe em 16 bits
70000 cabe em 32 bits
```

Exemplo de execução 3:

```
Digite um valor inteiro: 5000000000
5000000000 não cabe em 16 bits
5000000000 não cabe em 32 bits
```