Primeiros Programas

Inicialização de variáveis
Conversão de tipos,
Operações Matemáticas,
Expressões relacionais,
Constantes

Sumário

- Inicialização de variáveis
- Conversão de tipos
- Operações matemáticas
- Operadores Relacionais
- Declaração de constantes

Inicialização de variáveis

```
/* Programa “P1” */

#include <iostream>

using namespace std;

int main ()
{
   int nl;

   cout << nl << endl;

   cout << nl - 30 << endl;

   return 0;
}
```

- Que valores serão mostrados na saída?
  - **0** seguido de **-30** ?
  - A resposta é:
    - **Não se pode afirmar nada**
- Se programa não atribuir **explícitamente** valor a uma variável
  - variável tem valor **indeterminado**
    - **NUNCA** se pode afirmar qual o valor inicial de uma variável após sua declaração no programa

Inicialização de variáveis

Antes de ser usada, variável deve ser INICIALIZADA:

```
/* Programa "P1" */
#include <iostream>
using namespace std;

int main ()
{
  int nl = 0;

  float x, y;

  y = 0.5;

  cin >> x;

  cout << nl - 30 << " " << y * x << endl;

  return 0;
}
```

No momento da declaração

Com uma sentença de atribuição

Com entrada de dados

Conversão **float** ↔ **int**: atribuições

```
/* Programa “P1” */

#include <iostream>

using namespace std;

int main ()
{
   int nI;
   float nR;

   nI = 2.5;
   nR = nI;
   cout << nR << endl;
   return 0;
}
```

```
/* Programa “P2” */

#include <iostream>

using namespace std;

int main ()
{
   int nI;
   float nR;

   nR = 2.5;
   nI = nR;
   cout << nI << endl;
   return 0;
}
```

Pergunta: Qual resultado exibido nos dois programas?

Conversão **float** ↔ **int**: atribuições

```
/* Programa "P1" */

#include <iostream>

using namespace std;

int main ()
{
    int nI;
    float nR;

1   nI = 2.5;

2   nR = nI;
    cout << nR << endl;
    return 0;
}
```

Valor exibido: 2

1. Valor 2.5 → truncamento → 2 → nI
2. Valor 2 → converte para float → 2.0 → nR

```
/* Programa "P2" */

#include <iostream>

using namespace std;

int main ()
{
    int nI;
    float nR;

1   nR = 2.5;

2   nI = nR;
    cout << nI << endl;
    return 0;
}
```

Valor exibido: 2

1. Valor 2.5 → nR
2. Valor 2.5 → truncamento → 2 → nI

Conversão **float** ↔ **int**: expressões

```
/* Programa “P3” */

#include <iostream>

using namespace std;

int main ()
{
   int nI;
   float nR;

   nI = 2;
   nR = nI / 3;
   cout << nR << endl;
   return 0;
}
```

```
/* Programa “P4” */

#include <iostream>

using namespace std;

int main ()
{
   int nI;
   float nR;

   nR = 2.5;
   nI = nR / 2;
   cout << nI << endl;
   return 0;
}
```

Pergunta: Qual resultado exibido nos dois programas?

Conversão **float** ↔ **int**: expressões

```
/* Programa "P3" */

#include <iostream>

using namespace std;

int main ()
{
    int nI;
    float nR;

    nI = 2;

2   nR = nI / 3;
    cout << nR << endl;
    return 0;
}
```

1

Valor exibido: 0 (zero)

1 2 / 3 = 0.6667 → truncamento → 0
2 Valor 0 → converte para float → 0.0 → nR

```
/* Programa "P4" */

#include <iostream>

using namespace std;

int main ()
{
    int nI;
    float nR;

    nR = 2.5;

2   nI = nR / 2;
    cout << nI << endl;
    return 0;
}
```

1

Valor exibido: 1

1 2.5 / 2 = 1.25 → sem truncamento
2 Valor 1.25 → truncamento → 1 → nI

Conversão de Tipos

→ **Em atribuição:** tipo do valor à direita é convertido ao tipo da variável que recebe o valor. Ao atribuir **float** a um **int**, ocorre truncamento da parte decimal

→ **Em expressão:** o tipo do valor final da expressão é o tipo com mais "prioridade" dentre os valores na expressão

→ ***Prioridade de tipos:***

double ← float ← int ← char **[menor prioridade]**

Operações matemáticas

```
/* Programa “P5” */

#include <iostream>
#include <cmath>

using namespace std;

int main ()
{
   float nR, res;

   cin >> nR;
   res =  log(nR);
   cout << res << endl;
   return 0;
}
```

Operações mais complexas:

- **Necessário chamar funções matemáticas da biblioteca de funções *cmath***
- Essas funções recebem valores reais e devolvem um valor real como resultado

Exemplo: função log chamada no programa para calcular o logarítmo natural da variável nR

Operações matemáticas

```
/* Programa "P5" */

#include <iostream>
#include <cmath>

using namespace std;

int main ()
{
   float x, res;

   cin >> x;
   res =  log(x);
   cout << res << endl;
   return 0;
}
```

Funções matemáticas mais comuns da biblioteca *cmath*

- ***round** (x): calcula valor arredondado de **x***
- ***sqrt** (x):* $\sqrt{x}$
- ***cbrt** (x):* $\sqrt[3]{x}$
- ***pow** (base, exp):* $base^{\exp}$
- ***log** (x):* $\ln(x)$
- ***log10** (x):* $\log(x)$
- ***exp** (x):* e^{x}
- ***cos** (x), **tan** (x), **sin** (x): **x** → ângulo em radianos*
- ***fabs** (x): valor absoluto do número **real x***

Operações com tipo **char**.

```
/* Programa “maiuscula” */

#include <iostream>
#include <cctype>

using namespace std;

int main ()
{
   char letra, maiusc;

   cin >> letra;

   maiusc = toupper(letra);

   cout << maiusc << endl;
   return 0;
}
```

- **toupper(***letra***)**: transforma *letra* em maiúscula
- **tolower(***letra***)**: transforma *letra* em minúscula

Problema de leitura de **char**

```
/* Programa “charInt” */

#include <iostream>

using namespace std;

int main ()
{
   int idade;
   char letra;

   cin >> idade >> letra;

   cout << “idade: “ << idade << endl;
   cout << “letra: “ << letra << endl;
   return 0;
}
```

Qual o problema?

Espaços em branco, TAB e ENTER são usados por *cin*

Por enquanto nossos programas só farão leitura de valores numéricos.

Constantes

Valores fixos que não precisam estar em memória e não serão alterados durante o programa

Definição da constante.
Convenção: Todas as letras do identificador em MAIÚSCULA

Uso da constante

Uma constante NÃO pode estar no lado esquerdo de uma **atribuição**:

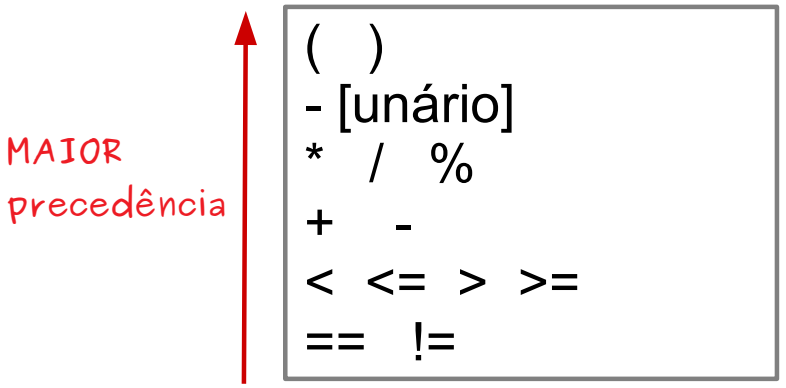
PI = 3.1416 → **ERRO !!!**

```
/* Programa "circulo" */

#include <iostream>

using namespace std;

#define PI 3.141592

int main ()
{
   float raio, circunf, area;

   cin >> raio;
   circunf = 2 * PI * raio;
   area = PI * raio * raio;
   cout << "circunferencia:  " << circunf << endl;
   cout << "area: " << area << endl;
   return 0;
}
```

Operadores Relacionais

Operadores Relacionais	
==	igual
!=	diferente
<	menor
<=	menor ou igual
>	maior
>=	maior ou igual

Precedência de operadores:



Valor de uma expressão relacional é **0** (Falso) ou **1** (Verdadeiro)

Exemplo de expressões relacionais

```
/* Programa “adulto” */

#include <iostream>

using namespace std;

int main ()
{
   int idade;

   cin >> idade;
   cout << “eh  adulto:  “ <<  (idade >= 18)  << endl;
   cout << “eh adulto idoso:  “ <<  (idade + 10 > 70)  << endl;
   return 0;
}
```

Erros comuns

- `idade = 20`
  - Esta expressão é RELACIONAL?
    - Ela verifica se valor de **idade** é igual ao valor **20**?
  - R: **não.** Esta expressão está ATRIBUINDO **20** à variável **idade**.
  - Para comparar: `idade == 20`

Erros comuns

- `11 < idade < 18`
  - Esta expressão quer verificar se idade está entre 11 e 18.
    - ➔ Se valor de **idade** é **10**, a expressão deveria ser FALSA (é o que se espera)
      - Ela é avaliada como VERDADEIRA !!!!

        (11 < idade) < 18 → 0 < 18 → VERDADEIRO

      - Queremos que:

        `11 < idade` seja VERDADE

        E

        `idade < 18` **também** seja VERDADE

Solução: Operadores LÓGICOS → **Próxima Semana**

Exercícios para aula *online*

Após assistir todas as vídeo-aulas da semana, procure trabalhar na **Lista de exercícios** do Tópico **Primeiros Programas**, na sala virtual da disciplina na UFPR Virtual.

Estes exercícios serão usados nas aulas *online* para esclarecer e consolidar os conceitos abordados até aqui.

Leitura complementar

Acesse o **Material complementar** do Tópico **Primeiros Programas**, na sala virtual da disciplina da UFPR Virtual.

Elas são importantes e auxiliam na compreensão dos temas abordados até aqui.

Créditos: O conteúdo original deste documento é de autoria da Profª Carmem Satie Hara (DINF/ET), e foi adaptado pelo Prof. Armando L.N. Delgado (DINF/ET) para uso na disciplina *Programação de Computadores* (CI208, CI180, CI183)

Compartilhe este documento de acordo com a licença abaixo