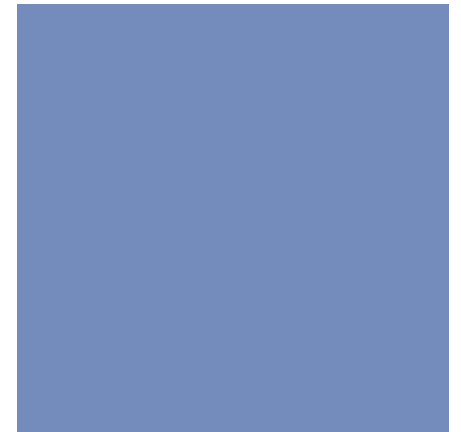
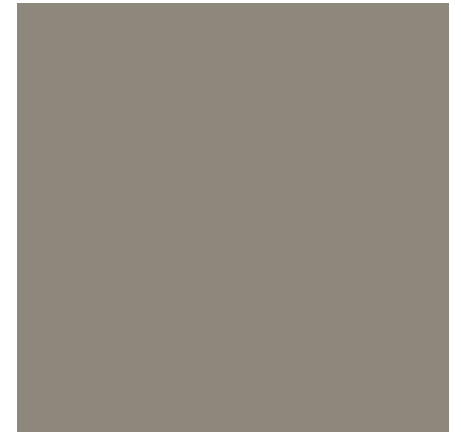




Entrada

Saída

Formatação



INF110 – Programação I

Prof. Alcione/André Gustavo
DPI/UFV – 2020/1



+ Em aulas anteriores...

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5
6     int prova1, prova2, prova3, notafinal;
7
8     //Ler nota da prova 1, prova 2, prova 3
9     cout << "Digite sua nota na prova 1: ";
10    cin >> prova1;
11    cout << "Digite sua nota na prova 2: ";
12    cin >> prova2;
13    cout << "Digite sua nota na prova 3: ";
14    cin >> prova3;
15
16    //Calcular a nota final: somar as 3 notas
17    notafinal = prova1 + prova2 + prova3 ;
18
19    //Escrever a nota final
20    cout << "Sua nota final foi ";
21    cout << notafinal << endl ;
22
23    return 0;
24 }
```

Entrada

Processamento

Saída



Entrada – cin

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

■ Sintaxe

- `cin >> variável;`

■ Funcionamento

- O programa interrompe a execução e aguarda entrada de dados
- Só avança quando for digitado um dado e pressionado 'enter'

■ Pode ser lida uma lista de dados

- `cin >> variávelA >> variávelB >> variávelC;`

■ Funcionamento

- Os dados são armazenados nas variáveis na ordem digitada
- Só avança após serem lidos todos os dados

+ Saída – cout

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

■ Sintaxe

- `cout << expressão;`

■ Funcionamento

- Avalia a expressão e escreve seu valor, de acordo com seu tipo
- Se for uma variável, escreve seu valor
- Se for uma expressão aritmética, avaliar e escreve o resultado
- Se for um texto entre aspas (" "), escreve o texto

■ Pode ser escrita uma lista de expressões

- `cout << expressãoA << expressãoB << expressãoC;`

+ Pronúncia

- `cout`
 - character **out**put
 - “cê-out”
- `cin`
 - character **in**put
 - “cê-in”

+ Saída – endl

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

- Sintaxe

- `cout << endl;`

- Funcionamento

- Usado na saída com `cout`
 - Encerra uma linha (o próximo dado sairá na linha seguinte)

- Pode ser usado dentro de uma lista de expressões

- `cout << expressãoA << endl << expressãoB << endl;`

+ Saída – ‘\n’

- Caracter especial, indica mudança de linha
 - Mesmo efeito que `endl`, mas usado dentro de texto entre “ ”

- Exemplo:

```
cout << "==UFV==\nBem-vindo!\nDigite sua matricula: ";
```

- Resultado:

```
==UFV==
Bem-vindo!
Digite sua matricula:
```

- Compare com a versão usando `endl`:

```
cout << "==UFV==" << endl << "Bem-vindo!" << endl << "Digite sua matricula: ";
```



Formatação de saída (pt. flutuante)

```
#include <iomanip>
```

- `cout << x;`
 - Escreve x com até 6 dígitos
 - Se for muito grande (ou muito pequeno), usa notação científica
- `cout << setprecision(2) << x;`
 - Escreve x com até 2 dígitos
 - Se tiver mais, arredonda
- `cout << fixed << setprecision(2) << x;`
 - Escreve x com exatamente 2 casas decimais
 - Se tiver mais, arredonda
 - Se tiver menos, preenche com 0


```

1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  int main()
6  {
7      double a = 3;
8      double b = 3.1;
9      double c = 3.14;
10     double d = 3.14159265358979;
11     double e = 0.0000027828;
12     double f = 2782800000;
13
14     cout << a << " " << b << " " << c << " " << d << " " << e << " " << f << endl;
15
16     return 0;
17 }

```

- Resultado:
 - 3 3.1 3.14 3.14159 2.7828e-006 2.7828e+009
 - No máximo 6 dígitos, em notação exponencial quando precisar mais que isso
- Resultado com `cout << setprecision(2) << a << ...`
 - 3 3.1 3.1 3.1 2.8e-006 2.8e+009
 - No máximo 2 dígitos
- Resultado com `cout << fixed << setprecision(2) << a << ...`
 - 3.00 3.10 3.14 3.14 0.00 2782800000.00
 - Todos com 2 casas decimais



Entrada e saída em C



- Em C++ usamos os objetos `cin` e `cout` da `iostream`
- Em C eram usadas as funções `scanf` e `printf` da `stdio`
 - `printf`: **print** formatted
 - `scanf`: **scan** formatted
- Para usar `printf` e `scanf` devemos informar o formato
- Em `cin` e `cout` o “formato” é automático pelo tipo da expressão
- Podem ser usados num mesmo programa, a preferência por um ou outro depende do contexto e de cada programador
- Para usar `printf` e `scanf` em C++, incluir a biblioteca `cstdio`



Entrada – scanf

```
#include <stdio>
```

■ Sintaxe

- `scanf ("%EspecificadorDeFormato", &variável) ;`

■ Funcionamento

- O programa interrompe a execução e aguarda entrada de dados
- Só avança quando for digitado um dado e pressionado 'enter'
- O dado é lido **de acordo com o formato especificado** e então armazenado na variável

■ Exemplo `scanf ("%d", &prova1) ;`

- `%d` indica que deve ser lido um inteiro na base decimal
- O `&` antes da variável obtém seu endereço (detalhes futuramente)



Entrada – scanf

```
#include <stdio>
```

■ Outros exemplos:

```
scanf("%d%d%d", &prova1, &prova2, &prova3);
```

- Lê 3 valores inteiros na base decimal, um para cada variável
- Os dados da entrada devem ser separados por espaços (ou enter)

```
scanf("%d/%d/%d", &dia, &mes, &ano);
```

- Lê 3 valores inteiros na base decimal, um para cada variável
- Os dados devem ser separados por '/', já que ela está no formato!

+ Saída – printf

```
#include <stdio>
```

■ Sintaxe

- `printf("Formato", ListaDeArgumentos);`

■ Funcionamento

- O programa escreve o texto indicado no Formato
- Se houver um especificador de formato (%Especificador) o valor seguinte da ListaDeArgumentos é formatado e escrito

■ Exemplo

```
printf("Nota final: %d", notafinal);
```

- Será escrita a mensagem `Nota final:` seguida de um espaço e o valor da variável `notafinal`, que substitui o especificador de formato `%d`.



Saída – printf

```
#include <stdio>
```

- Outro exemplo

```
printf("%d+%d+%d=%d", prova1, prova2, prova3, notafinal);
```

- Se as variáveis valem respectivamente 30, 20, 15, 65, a saída será:

30+20+15=65

- Compare com a versão usando cout

```
cout << prova1 << "+" << prova2 << "+" << prova3 << "=" << notafinal;
```



Saída – printf

```
#include <stdio>
```

- Outros especificadores de formato
 - %d – número inteiro decimal (signed int)
 - %u – número inteiro decimal unsigned
 - %f – número de ponto flutuante (float)
 - %l – long int
 - %ll – long long int
 - %lf – double
 - %c – caractere (char)
 - ...



Formatação de saída – printf

```
#include <stdio>
```

- Formatação
 - % [Tamanho] . [Precisão] Especificador
- **Tamanho:** indica o número mínimo de caracteres na saída
 - Se o valor a ser escrito for menor, acrescenta espaços em branco
 - Se for maior, não haverá modificação
- **Precisão:** indica a precisão do valor numérico
 - Se for de ponto flutuante, indica o número de casas decimais
 - Se for inteiro, número de dígitos (põe zeros à esquerda, se necessário)



Formatação ponto flutuante

```
1  #include <stdio>
2
3  int main()
4  {
5      double a = 3;
6      double b = 3.1;
7      double c = 3.14;
8      double d = 3.14159265358979;
9      double e = 0.0000027828;
10     double f = 2782800000;
11
12     printf("%.2lf %.2lf %.2lf %.2lf %.2lf %.2lf\n",a,b,c,d,e,f);
13
14     return 0;
15 }
```

■ Resultado

- 3.00 3.10 3.14 3.14 0.00 2782800000.00
- Todos com 2 casas decimais



Formatação inteiros

```
1  #include <stdio>
2
3  int main()
4  {
5      int a = 1;
6      int b = 12;
7      int c = 123;
8      int d = 1234;
9      int e = 12345;
10
11     printf("%d\n%d\n%d\n%d\n%d\n",a,b,c,d,e);
12
13     return 0;
14 }
```

■ Resultado

1
12
123
1234
12345

■ printf("%4d%4d...

1
12
123
1234
12345

■ printf("%.4d%.4d...

0001
0012
0123
1234
12345