

# **Fundação Hermínio Ometto**

## **Bacharelado em Sistemas de Informação**

SIF009 - Linguagem de Programação I

Prof. Dr. Sérgio Luis Antonello

**Aula 05**

01/09/020

# Plano de ensino

1. Unidade I - Programação estruturada e Linguagem C (objetivos b, c e d).
  - 1.1. Conceitos de programação estruturada.
  - 1.2. Estrutura de um programa de computador.
  - 1.3. Códigos fonte, objeto e executável.
  - 1.4. Biblioteca de códigos.
  - 1.5. Compiladores e Interpretadores.
  - 1.6. Processos de compilação e link edição.
  - 1.7. Identificação dos tipos de erros e alertas (léxicos, sintáticos e semânticos).
  - 1.8. Depuração de código.
  - 1.9. Palavras reservadas.
  - 1.10. Tipos de dados.
  - 1.11. Constantes. Variáveis simples e estruturadas. Escopo de variáveis.
  - 1.12. Operadores e precedência.
  - 1.13. Expressões aritméticas, lógicas e relacionais.
  - 1.14. Comandos.
  - 1.15. Ambientes de desenvolvimento e programação.
2. Unidade II - Estruturas de controle (sequência, decisão e repetição), registro e arquivo (objetivos a, c, d).
  - 2.1. Comandos if e switch.
  - 2.2. Comandos for, while e do-while.
  - 2.3. Blocos de comandos e aninhamento.
  - 2.4. Definição de tipos.
  - 2.5. Registro.
  - 2.6. Arquivo: leitura e gravação de dados em disco.
3. Unidade III - Ponteiros e Funções (objetivos a, c, d, e).
  - 3.1. Ponteiros.
  - 3.2. Funções.
  - 3.3. Passagem de parâmetro por valor.
  - 3.4. Passagem de parâmetro por referência.
4. Unidade IV- Strings e Variáveis indexadas (objetivos a, c, d).
  - 4.1. Manipulação de strings.
  - 4.2. Manipulação de caracteres.
  - 4.3. Declaração e manipulação de vetores.
  - 4.4. Declaração e manipulação de matrizes.

# Plano de ensino

Data	Atividade
04/08	Aula 01
11/08	Aula 02
18/08	Aula 03
25/08	Aula 04
01/09	Aula 05
08/09	Aula 06
15/09	Prova 1
22/09	Aula 08
29/09	Aula 09
06/10	Aula 10

Data	Atividade
13/10	Aula 11
20/10	Maratona FHO de Programação
27/10	Aula 13
03/11	Aula 14
10/11	Aula 15
17/11	Prova 2 Entrega Trabalho
24/11	Prova SUB
01/12	Semana Científica
08/12	Aula 19
15/12	Aula 20

# Sumário da aula

---

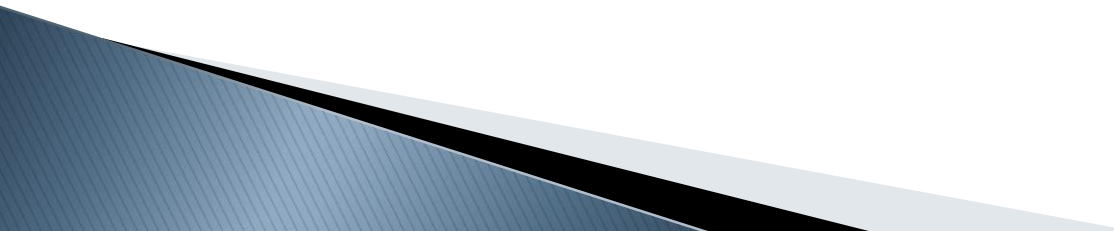
Primeiro momento (revisão)

- ✓ Estrutura de repetição – comando for.

Segundo momento (conteúdo)

- ✓ Estruturas de repetição
  - ✓ while
  - ✓ do-while

Terceiro momento (síntese)

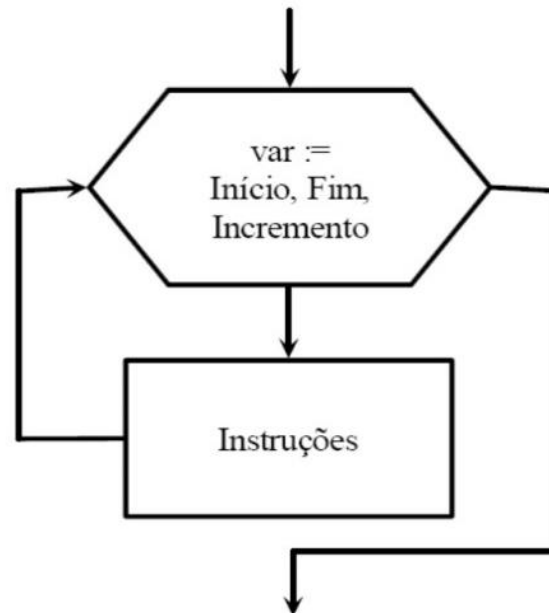
- ✓ Retome pontos importantes da aula
- 

# 1. Primeiro momento: revisão

Sintaxe do comando for:

```
for (inicialização; condição; incremento) {  
    comandos;  
}
```

Fluxograma:



# 1. Primeiro momento: revisão

---

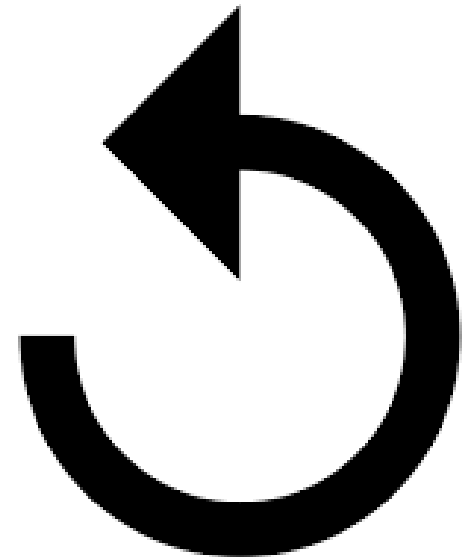
Exemplo:

```
for(i=0; i < 100; i++) {  
    if(i%2 == 0){  
        printf("%d eh um numero par", i);  
    }  
}
```

## 2. Segundo momento

---

- ▶ Estruturas de repetição



## 2. Segundo momento

---

- ▶ Estruturas de repetição
  - while
  - do while



# 3. Estrutura de repetição (loop)

Um loop é um recurso usado para repetir um comando ou bloco de comandos, tantas vezes quantas forem necessárias, para a resolução de um problema específico dentro do algoritmo.

Cada execução do loop é conhecida como laço ou laçada .

Podem ser classificados como:

- Laços contados : comando for.
- Laços condicionais: comandos **while** e **do-while** .

# 4. Comando while

---

O comando while é classificado loop condicional. O loop depende do resultado verdadeiro de uma condição para ser executado.

Sintaxe:

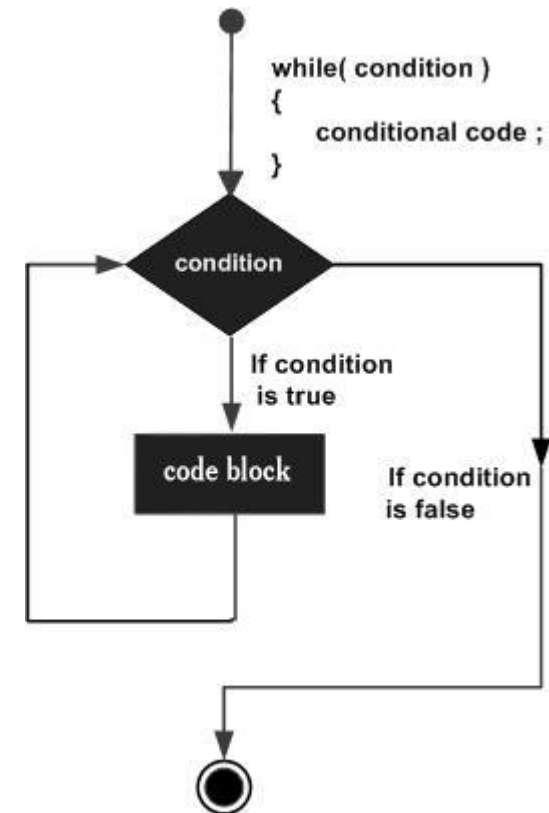
```
while (expressão)
{
    comandos a serem executados;
}
```

# 4. Comando while

Sintaxe:

```
while (expressão) {  
    um comando;  
    ou bloco de comandos;  
}
```

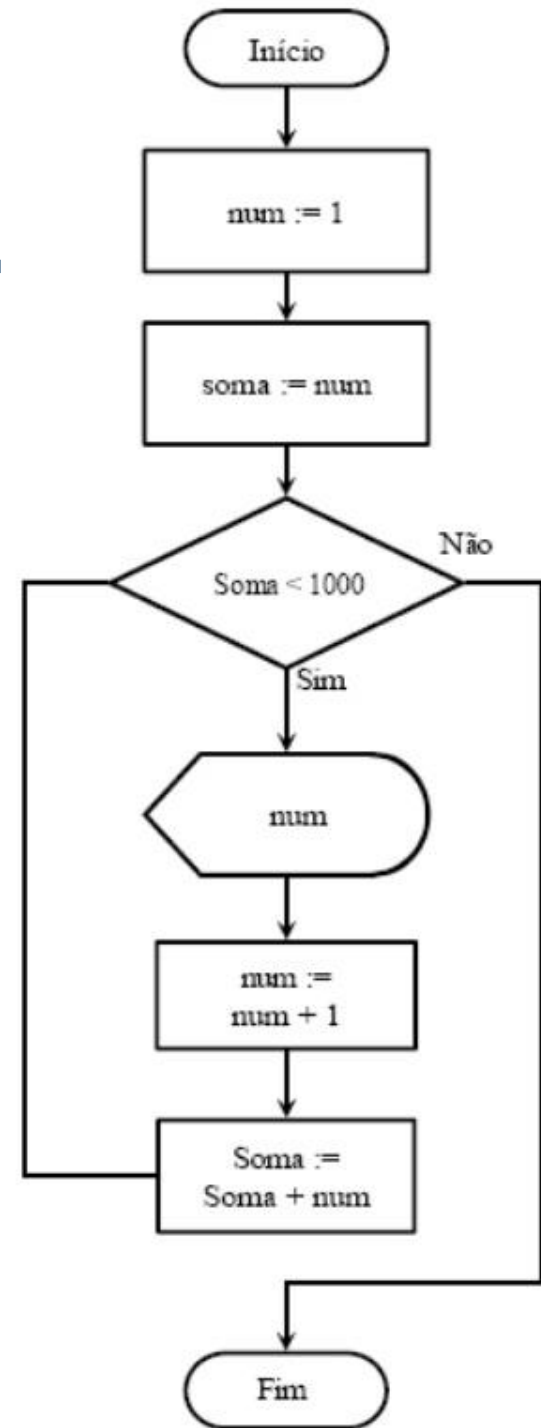
- Enquanto a expressão resultar em verdadeiro, a laçada é realizada e todos os comandos dentro do laço são executados.
- Ao término de uma laçada o controle volta a linha do while e a expressão é testada novamente.



# 4. Comando while

Exemplo: Executa a soma de valores enquanto o total for menor que 100.

```
int main () {  
    int num, soma;  
  
    num = 1;  
    soma = num;  
    while (soma < 100) {  
        printf("valor somado: %d\n",  
num);  
        num++;  
        soma = soma + num;  
    }  
    printf("Total: %d", soma);  
    return 0;  
}
```



# 5. Comando do-while

---

O comando do-while é classificado loop condicional. O loop depende do resultado de uma expressão para sair da laçada.

Sintaxe:

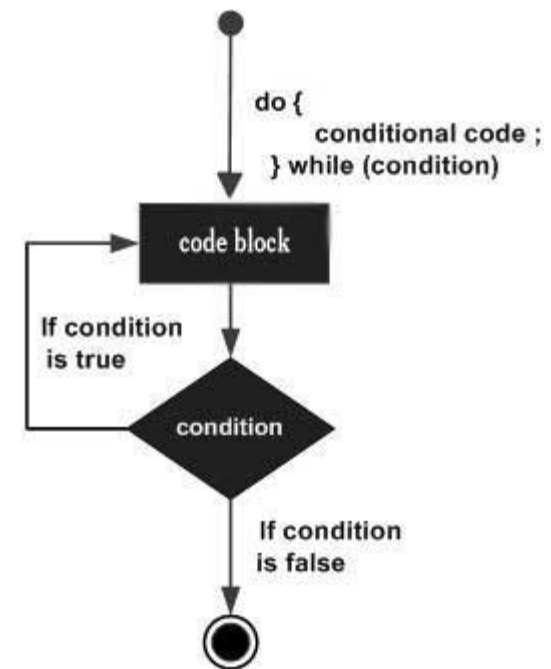
```
do {  
    comando;  
    ou bloco de comandos;  
  
} while (expressão);
```

# 5. Comando do-while

Sintaxe:

```
do {  
    comando;  
    ou bloco de comandos;  
} while (expressão);
```

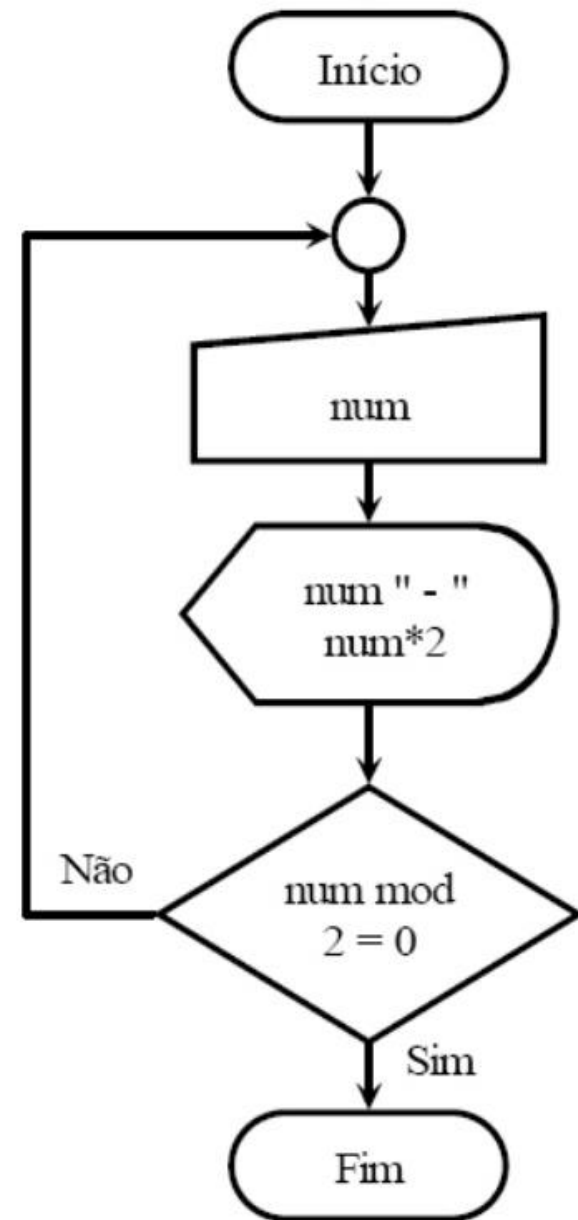
- Uma laçada é executada.
- Ao término de uma laçada a expressão é testada.
- No teste, se a condição resultar em verdadeiro, uma nova laçada é executada.



# 5. Comando do-while

Exemplo: Executa o laço até que seja entrado um número ímpar.

```
int main () {  
    int num;  
  
    do {  
        scanf("%d", &num);  
        printf("%d - %d\n", num, num*2);  
  
    } while (num % 2 == 0);  
  
    printf("Encerra o loop após ser  
    digitado um número ímpar");  
    return 0;  
}
```



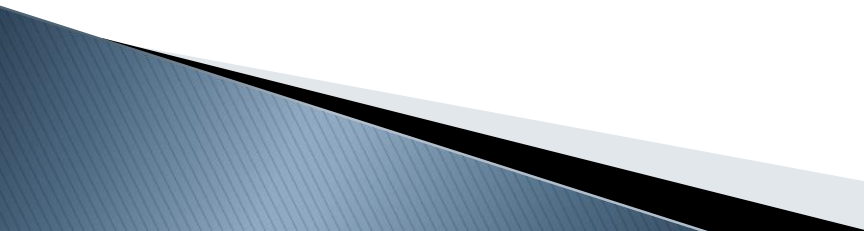
# 6. Comandos **break** e **continue**

---

O comando **break** faz com que o loop seja interrompido e a execução do programa vai continuar no próximo comando após o loop ou bloco que está sendo interrompido.

O comando **continue** funciona dentro de um loop e é útil quando uma parte da estrutura do loop deve ser desconsiderada por alguma condição específica.

Quando o **continue** é encontrado, a execução do programa pula para a próxima iteração do comando de loop, sem que o mesmo seja encerrado.





# 7. Loop & input

Em alguns casos a entrada de dados pode ser utilizada dentro da expressão que testa a condição de entrada no loop.

```
while (scanf ("%d %d", &a, &b) == 2)
```

```
while (scanf ("%d", &a) == 1)
```

```
while (scanf ("%d", &a) > 0 )
```

```
while (scanf ("%d", &a) && a == 1)
```

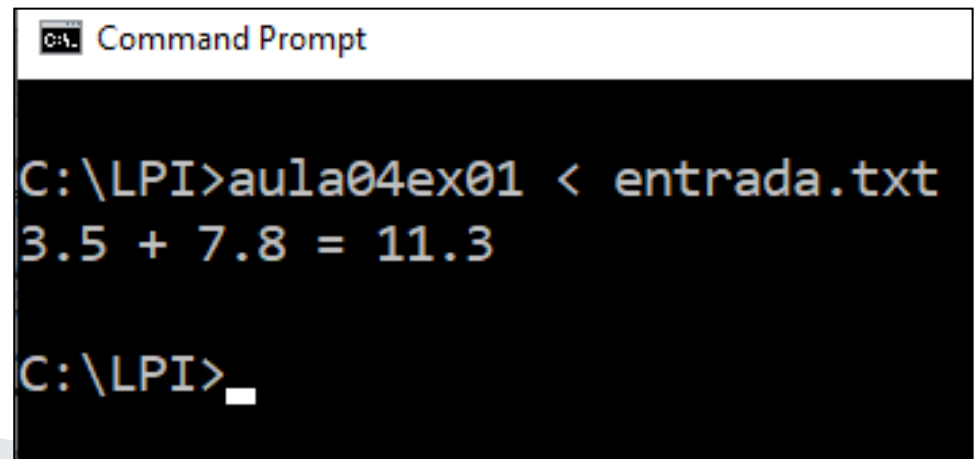
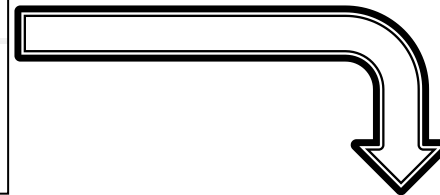
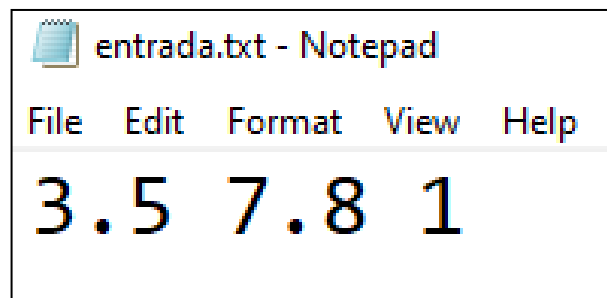
```
while (scanf ("%d", &a), a)
```

```
while ( scanf ("%d", &a) != EOF )
```

```
while (scanf ("%d %d", &a, &b) != EOF)
```

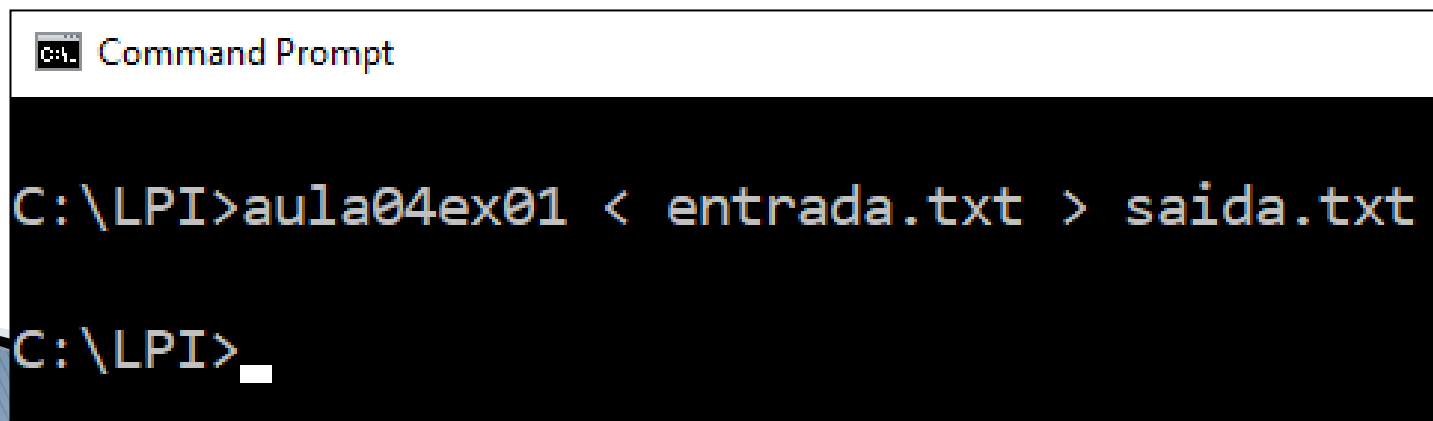
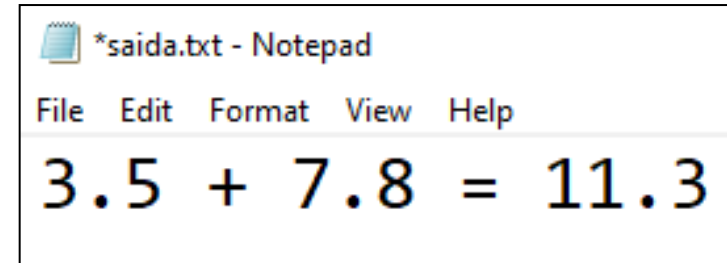
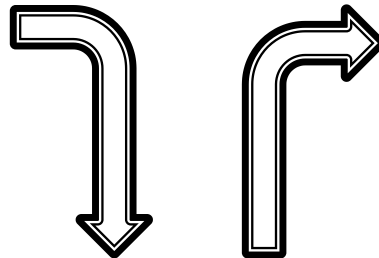
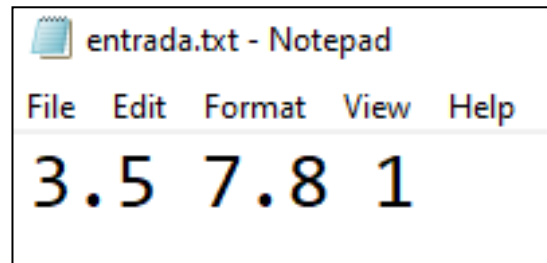
# 8. Redirecionamento de I/O

Para facilitar testes dos programas, é possível substituir a digitação de dados no teclado por dados gravados em um arquivo. Para isso, usa-se a execução do programa no prompt de comandos.



# 8. Redirecionamento de I/O

Da mesma forma, é possível destinar a saída de dados para um arquivo ao invés do monitor de vídeo.



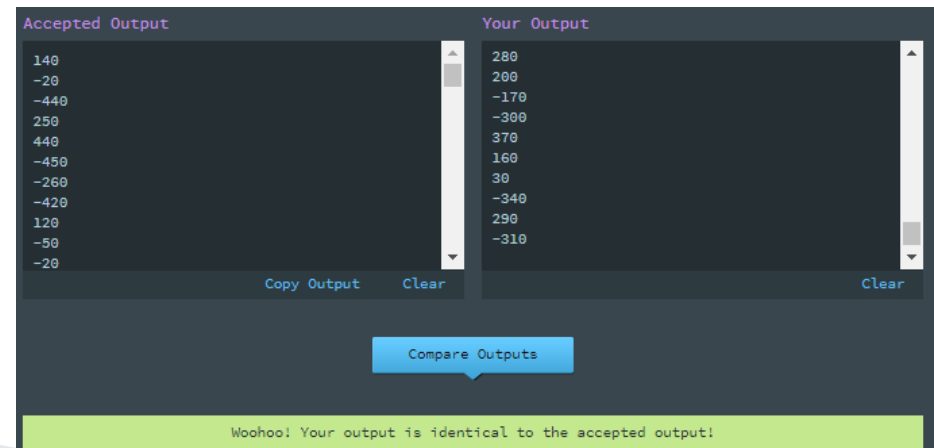
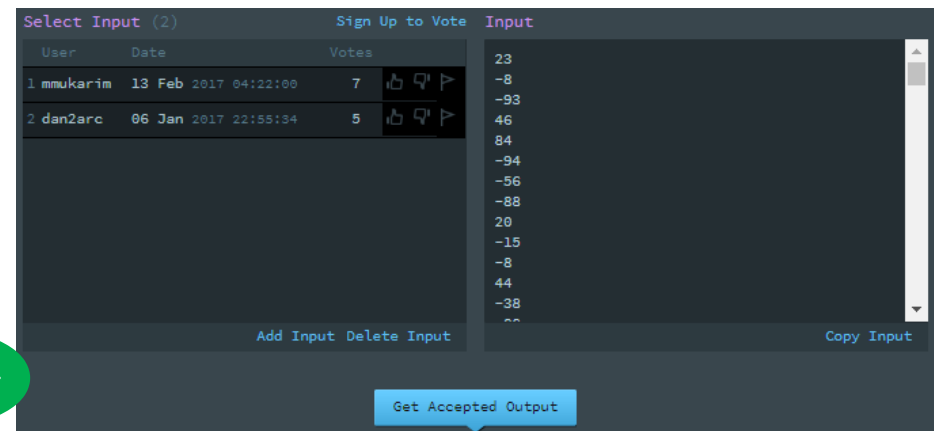
```
Command Prompt

C:\LPI>aula04ex01 < entrada.txt > saida.txt

C:\LPI>
```

# 9. UDebug

O Udebug é um software que apresenta dados de teste para validação de problemas do URI Online Judge.



# 10. Exercícios

---

**Vamos  
Programar!**



# 10. Atividade em grupo: aluno fala e professor digita

## 1) URI1159 Soma de Pares Consecutivos

<https://www.urionlinejudge.com.br/judge/pt/problems/view/1159>

O programa deve ler um valor inteiro **X** indefinidas vezes. (O programa irá parar quando o valor de **X** for igual a 0). Para cada **X** lido, imprima a soma dos 5 pares consecutivos a partir de **X**, inclusive o **X**, se for par. Se o valor de entrada for 4, por exemplo, a saída deve ser 40, que é o resultado da operação:  $4+6+8+10+12$ , enquanto que se o valor de entrada for 11, por exemplo, a saída deve ser 80, que é a soma de  $12+14+16+18+20$ .

### Entrada

O arquivo de entrada contém muitos valores inteiros. O último valor do arquivo é zero.

### Saída

Imprima a saída conforme a explicação acima e o exemplo abaixo.

Exemplo de Entrada	Exemplo de Saída
4 11 0	40 80

# 10. Atividade em grupo: aluno fala e professor digita

Soma 5 números pares consecutivos

## Exemplo de Entrada

4

11

0

$4+6+8+10+12$

$12+14+16+18+20$

## Exemplo de Saída

40

80

# 10. Atividade em grupo: aluno fala e professor digita

---

- 1) URI1159 Soma de Pares Consecutivos
  - a) Ler o enunciado e abstrair a solução
  - b) Desligar todos os monitores
  - c) Codificar em linguagem C da seguinte forma:
    - ✓ Um aluno fala um comando (ou linha)
    - ✓ O professor digita
    - ✓ Repete-se as etapas 1 e 2 até completar o código
  - d) Com dados do Udebug, executar com redirecionamento de I/O usando os arquivos "input.txt" e "output.txt"
  - e) Submeter a solução no URI
  - f) Subir o código na classroom (com status de accepted)



# 10. Exercícios

---

## 2) URI1146 Sequências Crescentes

<https://www.urionlinejudge.com.br/judge/pt/problems/view/1146>

## 3) URI1154 Idades

<https://www.urionlinejudge.com.br/judge/pt/problems/view/1154>

## 4) URI1117 Validação de Nota

<https://www.urionlinejudge.com.br/judge/pt/problems/view/1117>

## 5) URI1060 Números Positivos

<https://www.urionlinejudge.com.br/judge/pt/problems/view/1060>



# 11. Terceiro momento: síntese

---

- As estruturas de repetição possibilitam executar repetidas vezes um determinado trecho de código.
- Esse processo é chamado de loop.
- Na linguagem C, um loop pode ser escrito pelos comandos “for”, “while” e “do while”.
- O comando “for” possibilita contar quantas laçadas serão executadas.
- Os comandos “while” e “do while” usam o teste de uma condição para saber se iniciam ou continuam no loop.