



# INF029 - Laboratório de Programação

Prof. Renato Novais

INF029 – Laboratório de Programação  
renatonovais@gmail.com

# Agenda



- A Disciplina
- O Professor
- Método de Ensino
- Visão geral da Disciplina

# Agenda



- A Disciplina
- O Professor
- Método de Ensino
- Visão geral da Disciplina

# A disciplina: Objetivos Gerais

- **Continuar** o desenvolvimento dos conceitos básicos de lógica de programação, **estimulando** o **raciocínio lógico** e **estruturado** para resolver problemas e desenvolver algoritmos, **praticando** conceitos com uso de uma linguagem de programação estruturada.

# A disciplina: Ementa

- 1ª Parte
  - Modularização, funções, passagem de parâmetros por valor
- 2ª Parte
  - Ponteiros, passagem de parâmetros por referência, alocação dinâmica
- 3ª Parte
  - API'S (Streams), Recursividade

# A disciplina: Avaliação

- Para cada parte da disciplina é realizado pelo menos:
  - **Trabalhos** com valor de **3,0 pontos**, e
  - **Teste/Prova** escrita com valor de **7,0 pontos**
  - Essa pontuação pode sofrer ajustes a depender do andamento do semestre, a qual será informada previamente aos estudantes.
  - A prova é realizada no horário de aula
- Controle de Frequência diretamente no sistema acadêmico ou via lista: **necessário 75% de presença**.
  - Realizada após os 20 minutos de aula.

# A disciplina: Atendimento Extra Classe



- O IFBA dispõe de **horário de atendimento** para suas disciplinas.
- O Aluno pode agendar horário de atendimento com o professor, por email, nas quartas e sextas feiras do semestre entre 14h e 17:30h.
- O agendamento deve ser feito via email do professor: [renatonovais@gmail.com](mailto:renatonovais@gmail.com)
- Local: GSORT (Sala A 303)

# A disciplina: Canal de comunicação



- A disciplina dispõe um canal de comunicação via grupos de e-mails
  - [ifba\\_inf029@googlegroups.com](mailto:ifba_inf029@googlegroups.com), ao enviar um email para esse email, todos os cadastrados, inclusive o professor receberá a mensagem.
  - Recomenda-se que dúvidas sejam enviadas para o grupo para que outros possam se beneficiar também.
- Para se cadastrar no grupo, o aluno deve enviar um e-mail para o professor da disciplina na primeira semana de aula, com o assunto: INF029.



# A disciplina: Material de Apoio

- Site da disciplina: <http://ads.ifba.edu.br/INF029>
  - Neste link tem um conjunto inicial de materiais que deve ser utilizado para estudo. Há também algumas vídeo-aulas, e referências para material bibliográfico.
- Repositório da disciplina:  
<https://github.com/renatoIn/INF029>
  - Neste link tem vários materiais utilizados na disciplina, incluindo especificação dos trabalhos, exemplos de código, lista de exercícios, provas anteriores, etc.

# A disciplina: Ferramentas

- Editor de texto (ex: Sublime, kate)
- GCC
- Linux
- IDEs/Editores
  - Code Blocks
  - Visual Code Studio
    - Instalar plugins C/C++ e Code Runner (colocar suporte para rodar no terminal)

# A disciplina: *feedback* dos alunos

- Semestralmente os alunos fazem uma avaliação da disciplina. Alguns depoimentos são deixados por alunos de forma anônima, incluindo críticas, elogias e sugestões
- O objetivo é melhorar a disciplina a cada ciclo
- As avaliações estão disponíveis no site da disciplina
- Os alunos podem ler para ter uma ideia da visão dos alunos que já passaram por ela.

# Agenda



- A Disciplina
- O Professor
- Método de Ensino
- Visão geral da Disciplina

# O professor: Renato Novais



- Doutor em Ciência da Computação
- Currículo <http://lattes.cnpq.br/5036544358013553>
- Membro do GSORT (<http://gsort.ifba.edu.br>)
- Professor do PPGESP (<http://ppgesp.ifba.edu.br>)
- Pesquisa em Visualização de Informação e Análise de Dados e Desenvolvimento de aplicativos móveis
- E-mail: [renatonovais@gmail.com](mailto:renatonovais@gmail.com)
- Disciplinas na graduação: INF029 e INF022 – Tópicos Avançados

# Agenda



- A Disciplina
- O Professor
- Método de Ensino
- Visão geral da Disciplina

# Método de Ensino

- Focado no desenvolvimento da **autonomia** do aluno
  - Conforme *feedback* dos alunos, há críticas e elogios
- Parte do pressuposto que para adquirir as habilidades e competências **é preciso ter tempo fora da sala de aula para estudar**. Pelo menos **4h** semanais.
- **Disponibilidade para ajudar fora do horário de aula**, seja nos horários de atendimento (preferencialmente) ou, em alguns casos, virtualmente.

# Agenda



- A Disciplina
- O Professor
- Método de Ensino
- Visão geral da Disciplina



# Estruturas de dados: vetores

- **Vetores**: Estrutura de dados para guardar múltiplos valores do mesmo tipo
  - `int vet[10];`
  - `vet [4] = 30;`

# Estruturas de dados: Struct

- **Struct**: permite criar um tipo de dados que pode ser usado para agrupar itens de tipos possivelmente diferentes em um único tipo

```
typedef struct dma {  
    int dia;  
    int mes;  
    int ano;  
} Data;
```

```
/*Criando a struct aluno */  
typedef struct dados_aluno  
{  
    int matricula;  
    char nome[50];  
    char sexo; //M - Masculino, F - Feminino  
    Data data_nascimento;  
    char cpf[15];  
} Aluno;
```

# Estruturas de dados: Struct



Definindo a variável do tipo aluno e um vetor de Aluno

```
Aluno aluno; /*Criando a variável aluno que será do tipo struct Ficha_Aluno */  
Aluno lista_aluno[10]; // vetor para armazenar a lista de alunos
```

Usando o aluno

```
printf("\n### Cadastro de Aluno ###\n");  
printf("Digite a matrícula: ");  
scanf("%d", aluno.matricula);
```

Usando o vetor de aluno lista\_aluno

```
printf("\n### Cadastro de Aluno ###\n");  
printf("Digite a matrícula: ");  
scanf("%d", &lista_aluno[0].matricula);
```

# Modularização

- Tem como objetivo fazer a decomposição do código fonte em módulos

```
int main(){
    int x, y, a, b, resultado;
    scanf("%d", &x);
    scanf("%d", &y);
    resultado = x + y;
    printf("%d", resultado);

    scanf("%d", &a);
    scanf("%d", &b);
    resultado = a + b;
    printf("%d", resultado);

    return 0;
}
```

```
void soma();

int main(){
    soma();
    return 0;
}

void soma(){
    int x, y, a, b, resultado;
    scanf("%d", &x);
    scanf("%d", &y);
    resultado = x + y;
    printf("%d", resultado);

    scanf("%d", &a);
    scanf("%d", &b);
    resultado = a + b;
    printf("%d", resultado);
}
```

# Ponteiros

- Ponteiros são tipos de dados que referenciam (ou “apontam” para) endereços de memória
- Com ponteiros é possível:
  - Funções modificar seus argumentos
  - Fazer rotinas de alocação dinâmica
  - Aumentar a eficiência de certas rotinas

# Ponteiros

```
#include <stdio.h>
#include <stdlib.h>

int ponteirosStart()
{
    printf("Hello ponteiros!\n");
    int i = 2;
    int *p = &i;
    printf("Endereço de i: %p\n", p);
    printf("Valor de i %d\n", *p);
    //printf("%p\n", &ponteiro);
    return 0;
}
```

```
Hello ponteiros!
Endereço de i: 0x7fff57b63abc
Valor de i 2
```

# Arquivos

- Um arquivo pode estar associado a qualquer dispositivo de entrada e saída, como por exemplo: teclado, vídeo, impressora, disco rígido, etc.

```
#include<stdio.h>

int main(){
    FILE *arquivo; //vai ser associada ao arquivo
    arquivo = fopen("c:/luis/teste9.txt","r");
    if(arquivo==0)
        printf("Erro na leitura do arquivo\n");
    else
        printf("Arquivo aberto com sucesso\n");
    fclose(arquivo); //fecha arquivo
    system("pause");
}
```

# Recursão

- Função recursiva é aquela que chama a si própria.

```
int main(){  
    soma();  
    return 0;  
}
```

```
void soma(){  
    int x, y, a, b, resultado;  
    scanf("%d", &x);  
    scanf("%d", &y);  
    resultado = x + y;  
    printf("%d", resultado);  
  
    scanf("%d", &a);  
    scanf("%d", &b);  
    resultado = a + b;  
    printf("%d", resultado);  
    soma();  
}
```





# Dúvidas?

INF029 – Laboratório de Programação  
Prof. Renato Novais / [renatonovais@gmail.com](mailto:renatonovais@gmail.com)