gthc.c low_level.rs

```
GREEN TEAM HACKER CLUB {
    [Programação Low Level]

        < Introdução à Data Structures>

}
```

Tabela de ‘Conteúdo’ {

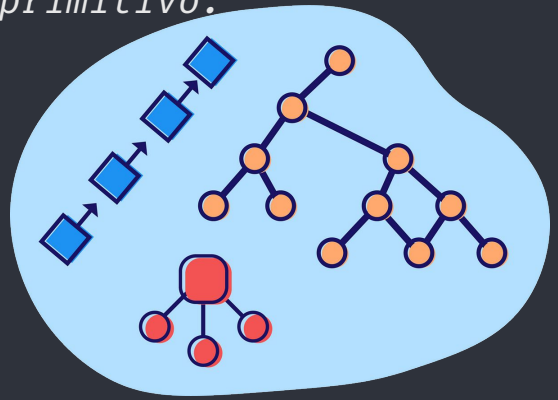
}

```
01 {
    [Estruturas de Dados]
}
```


Definição < /1 > {

Uma estrutura de dados (ED), em ciência da computação, é uma coleção tanto de valores (e seus relacionamentos) quanto de operações (sobre os valores e estruturas decorrentes). É uma implementação concreta de um tipo abstrato de dado (TAD) ou um tipo de dado (TD) básico ou primitivo.

}



Explicação < /2 > {

Em C e nas linguagens de baixo nível, umas das principais preocupações relacionadas à programação é o gerenciamento de dados. Isso pois a maneira como o os dados são manipulados está ligado diretamente com a eficiência do programa. Em geral, sempre queremos ocupar o mínimo de espaço (principalmente quando existe uma limitação no hardware) e queremos gastar o mínimo de tempo

}

Exemplo < /3 > {

Suponha que queremos criar um programa de baixo nível para armazenar os dados dos membros do GTHC. Queremos guardar seus nomes, informações, projetos. Mas qual seria a melhor forma de fazer isso pensando em limitações de espaço e tempo?

E se quisermos encontrar alguém específico, como fazer isso?

}

Aplicações < /4.1 > {

Estruturas de dados são usadas em quase todos os programas ou sistemas de software. Alguns exemplos comuns de.
estruturas de dados são arrays, listas vinculadas, filas, pilhas, árvores binárias e tabelas hash.
são amplamente aplicados nas seguintes áreas:

- *Projeto do compilador*
- *Sistema operacional*
- *Pacote de análise estatística*
- *SGBD*
- *Análise numérica*
- *Simulação*
- *Inteligência artificial*
- *Grafos*

}

Aplicações < /4.2 > {

Ao selecionar uma estrutura de dados para resolver um problema, as etapas a seguir devem ser executados:

1. *Análise do problema para determinar as operações básicas que devem ser suportadas. A operação básica pode incluir inserir/excluir/pesquisar um item de dados da estrutura de dados.*
2. *Quantifique as restrições de recursos para cada operação.*
3. *Selecione a estrutura de dados que melhor atenda a esses requisitos.*

}

Análise Assintótica< /5.1 > {

Notação Big-0
Nós calculamos o big-0 de um algoritmo contando o número de iterações que o algoritmo sempre faz com uma entrada de n.

Notação Big-O
A notação Big-O descreve o pior tempo de execução de um programa.

Notação Big-Ω
Big-Ω (Omega) descreve o melhor tempo de execução de um programa.

}

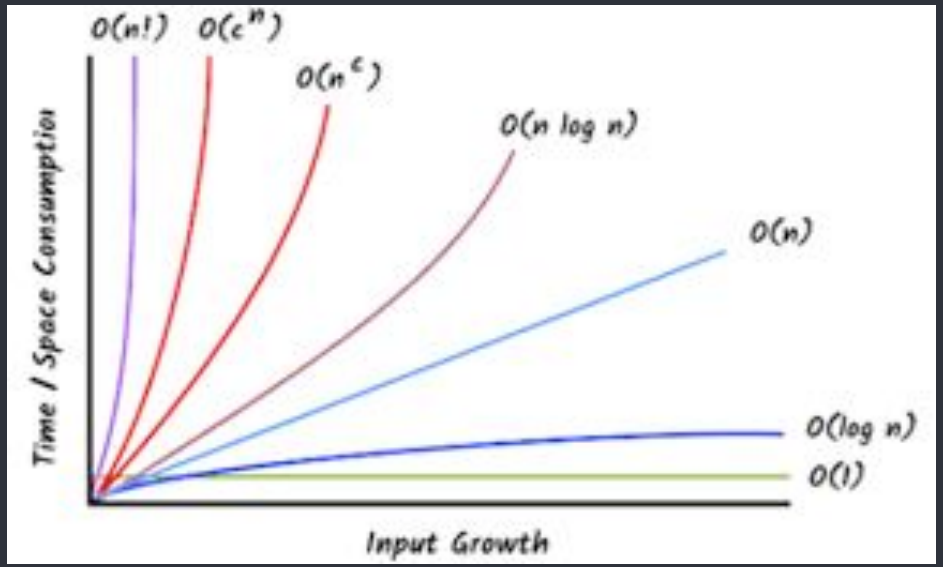
Análise Assintótica< /5.2 > {

Tempos de execução algorítmicos
Os tempos de execução algorítmicos comuns do mais rápido ao mais lento são:

constante: Θ(1)
logarítmico: Θ(log N)
linear: Θ(N)
polinomial: Θ(N^2)
exponencial: Θ(2^N)
fatorial: Θ(N!)

}

Análise Assintótica< /5.3 > {



}

Tipos < /6 > {

Primitivos e Não-primitivos
Floats, Arrays, Strings / Listas, Árvores

Lineares e Não Lineares
Arrays, Listas Ligadas, Pilhas / Grafos e Árvores

}

Algoritmos < /7 > {

Além das estruturas de dados, entender algoritmos de ordenação, classificação e armazenamento de dados é fundamental para o desenvolvimento de software

}

Linguagem de Máquina< /8 > {

Como as estruturas de dados são representadas em código de máquina?

Em geral, o código em binário e assembly não possui grandes alterações, o que altera é quantas chamadas são necessárias

}

Arrays < /9 > {

Está contido na maioria das linguagens, é útil como uma própria estrutura de linguagem e também para criar estruturas de linguagens.

Possui acesso em tempo linear, mas restrições de tamanho e ordenação

}

Arrays < /9 > {

```
#include <stdio.h>

// structure template
struct Employee {
    char Name[20];
    int employeeID;
    int WeekAttendence[7];
};

// driver code
int main()
{
    // defining array of structure of type
Employee
    struct Employee emp[5];

    // adding data
    for (int i = 0; i < 5; i++) {
        emp[i].employeeID = i;
        strcpy(emp[i].Name, "Amit");
        int week;
        for (week = 0; week < 7; week++) {
            int attendence;
            emp[i].WeekAttendence[week] = week;
        }
    }
```

```
    // printing data
    for (int i = 0; i < 5; i++) {
        printf("Emplyee ID: %d - Employee Name:
%s\n",
                emp[i].employeeID, emp[i].Name);
        printf("Attendence\n");
        int week;
        for (week = 0; week < 7; week++) {
            printf("%d ",
emp[i].WeekAttendence[week]);
        }
        printf("\n");
    }

    return 0;
}
```

}

Visualização {

<Data Structure Visualizations - https://www.cs.usfca.edu/~galles/visualization/Algorithms.html >

}

Referências {

<Data Structures Using C 2nd edition - https://github.com/GauravWalia19/Free-Algorithms-Books/blob/main/Library/src/C/Data-Structures-Using-C-2nd-edition.pdf >

}

Próximo Encontro {

< Listas >

}

```
Obrigado; {

    'Dúvidas?'

        luccas.h.cortes@hotmail.com
        https://github.com/Cortesz/

                CREDITS: This presentation template was
                created by Slidesgo, including icons by
                Flaticon, and infographics & images by Freepik

                < https://github.com/greenteamhc >

}
```