Curso de C

Fundamentos da computação

- Computar: Processamento de dados por meio de um sistema de I/O (input/output, entrada/saída).
- Software: Resumidamente, os programas/códigos.
- Hardware: Resumidamente, parte física do computador.
- Algoritmo: Sequencia finitas de instruções ordenadas.
- Programar: Processo onde o programador escreve um algoritmo se utilizando de uma linguagem de programação (no caso, C).
- IDE: Ambiente de desenvolvimento integrado.
- Código-fonte → Compilador → Programa.exe
- Variável: espaço na memoria que contem ou não um valor. Possuindo endereço na memoria, tipo (char, int, float, bool, etc), nome, e um valor (ou não).
- Função: Conjunto de comandos que realiza um função especifica. Possuindo tipo de retorno (podendo inclusive não possuir, assim sendo declarado como "void" [vazio]), nome, parâmetros de entrada, e código interno.
- Bibliotecas: Conjunto de funções já definidas previamente. Importante lembrar: stdio.h (biblioteca que permite o sistema de I/O) e stdlib (biblioteca para alocar memoria, gerar números aleatórios, converter variáveis, etc).

Origem do C

- O desenvolvimento ocorreu entre 1969 até 1973 (sendo o ano de 1972 o mais produtivo). C recebeu esse nome por ser derivado da linguagem B que, por sua vez, era uma versão reduzida da linguagem BCPL.

Alto e baixo nível

- Baixo nível: Mais próximo da linguagem de maquina (como assembly).
- Alto nível: Mais próximo da linguagem humana (como C, apesar dele poder ser considerado um "nível medio").

Tipos de variáveis

- Para usar acentos, usar a biblioteca locale.h e setlocale(LC_ALL, "") dentro da main().
- Comentários em c → //Alguma_coisa → /* Alguma coisa */
- Para escrever texto em c → printf("alguma_coisa") | Para escrever o valor dentro de uma variável inteira → prinf("%d",variável) → Obs: Pode misturar com texto, como → prinf("Alguma_coisa: %d", variável) | \n → Quebra de linha
- Ler o input do usuário → scanf("%d", &variável) → & = endereço na memoria.
- Inteiro → %d | Float → %f | Char(caractere, tem de ser em aspas simples → " → E é necessario limpar o buffer [por meio de fflush(stdin)] antes de ler algum caractere) → %c
- + → Adição | - → Subtração | * → Multiplicação | / → Divisão | % → Resto da divisão | abs() → Valor absoluto (resumidamente, transforma valores negativos em positivos).

Sobre valores boleanos

- Para usar valores booleanos (true or false) em C, incluir a biblioteca stdbool.

- Aliás, 0 → Falso | 1 → Verdadeiro

Como saber o valor em ASCII de um caractere

- Basta, ao imprimir, colocar %d ao invés de %c, e ele ira mostrar o valor correspondente do caractere em ASCII

Definir constantes em C

- #define *NOME *valor

Vetores em C (array's unidimensionais)

- Resumidamente um vetor (array unidimensional) é como um "lista" de n valores do mesmo tipo (que, em se tratando da memoria, se encontram em sequencia).
- Exemplo de sintaxe: tipo nome_vetor[quantidade_elementos] | nome_vetor[0] = valor_0 → nome_vetor[1] = valor_1 → nome_vetor[n] = valor_n

Strings em C

- Strings em c são arrays de caracteres (com um \0 no final), sendo %s para exibi-las. Antes de colocar um input, usar o setbuf(stdin, 0) para limpar o input, e no caso de strings, usar o fgets(nome_variavel_string, tamanho_string, meio_de_entrada [leia-se stdin]) → Isso evita possíveis buffer overflows.
- Para "liberar espaço na memoria" em uma string, utiliza-se array_string[strlen(array_string) - 1] = '\0' → O -1 é devido ao fato do \n (enter) ser também considerado um caractere. Por exemplo → string a[10] = "abc" | Isso ficaria algo como [a][b][c][\n][][][][][][\0] | O strlen lê o total de caracteres sem o "\0", então ele neste caso ele retornaria 4, contudo, como os index começam no zero, teriamos que colocar ele no index 3 (no caso, o \n), o que no final resulta em → [a][b][c][\0][][][][][][]

Strings em c++

- Utilizar as bibliotecas <string> e <iostream>
- sintaxe → std::string palavra; [poderia usar o using namespace std para colocar string direto]| cin >> palavra; | cout << palavra; → da pra concatenar em c++ usando o cout → cout << "alguma_coisa" << palavra

Matrizes em C

- Basicamente, uma matriz em C trata-se de um vetor bidimensional. Isso é, um vetor de vetores.
- Sintaxe basica → tipo_variavel variavel[tamanho_x][tamanho_y];

Ponteiros em C

- & → Endereço de memoria
- Variáveis armazenam valores, ponteiros armazenam posições na memoria
- Sintaxe: tipo_dado_variavel_apontado *ponteiro = &variável
- * → "Contudo apontado por" | Exemplo: int b = 40 → int *ponteiro = &b → *ponteiro = 20 → O valor de b passa a ser 20

Funções em C

- Sintaxe: tipo_retorno_função nome_função(tipo_do_parametro1 parametro1, tipo_do_parametro parametro2){ codigo_a_ser_executado}

Parâmetros em C

- Parâmetro = Valor que sera passado para dentro da função

Ponteiros como parâmetros em C

- Sintaxe → tipo_retorno nome_função(tipo_ponteiro *ponteiro) {codigo}
  Exemplo:

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

void aumentaDez(int *ptr_numero){
        *ptr_numero += 10;
}

void main(){
        int a = 10;
        printf("Número original: %d\n", a);
        aumentaDez(&a);
        printf("Novo valor: %d", a);
}
```

Vetores como parâmetros em C

- Conceito → Quanto um vetor é criado, ele é tratado como um ponteiro (então).
- Exemplo sintaxe:

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

void aumentaVetor(int *vetor, int tamanho){
        for(int x = 0; x < tamanho; x++){
        vetor[x] += 1;
      }
}

void main(){
        int vetor[3] = {1,2,3};
        for(int x = 0; x < 3; x++){
             printf("%d\n", vetor[x]);
          }
       aumentaVetor(vetor, 3);
       for(int x = 0; x < 3; x++){
             printf("%d\n", vetor[x]);
       }
}
```

Inserindo arquivo .h em C

- Basicamente, crie um arquivo com as funções desejadas, tendo .h como extensão, e utilize #include "nome_do_arquivo.h"

Alocação dinâmica de vetores em C

- Codigo exemplo:

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

void main(){
  int *ptr, tamanho_array;
  printf("Digite o tamanho do array: ");
  scanf("%d", &tamanho_array);
```

```
  ptr = malloc(tamanho_array * sizeof(int)); // Leia-se aloque ("posicione") na memoria a //quantidade do valor determinado pelo usuari
  for(int x=0; x < tamanho_memoria; x++){
      ptr[x] = x;
  }
  for(int x=0; x < tamanho_memoria; x++){
      printf("%d\n", ptr[x]);
  }
}
```

- Alias, após alocar memoria, utilizar a função free(vetor) para limpar a memoria.

Alocação dinâmica de matrizes em C

- Código exemplo:

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

void main(){
int **matriz, x_matriz = 3, y_matriz = 3, x, y; // **matriz = Basicamente, um ponteiro que aponta para um ponteiro
matriz = malloc(x_matriz * sizeof(int *));  // Leia-se, aloque o equivalente três (x_matriz) vezes o quanto ocupa na memoria um ponteir
for(x = 0; x < y_matriz; x++){
matriz[x] = malloc(y_matriz * sizeof(int));  // Leia-se, aloque na posição x o equivalente a três (y_matriz) vezes o quanto ocupa na me
}
for(x = 0; x < x_matriz; x++){
for(y = 0; y < y_matriz; y++){
matriz[x][y] = y;
}
}
for(x = 0; x < x_matriz; x++){
for(y = 0; y < y_matriz; y++){
printf("%d ", matriz[x][y]);
}
printf("\n");
}
}
```

Structs em C

- Basicamente, pense nisso quase que como se estivesse declarando uma classe, podendo acessar os objetos.
- Exemplo:

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

struct pessoa{
int idade;
char nome[255];
char profissao[255];
};
void main(){
struct pessoa primeira_pessoa;
primeira_pessoa.idade = 34;
strcpy(primeira_pessoa.nome, "Jorge");
strcpy(primeira_pessoa.profissao, "Marceneiro");
printf("O nome da primeira pessoa é %s, tendo %d anos, sendo sua profissão %s", primeira_pessoa.nome, primeira_pessoa.idade, primeira_p
}
```

Escrever em arquivos em C++

- Primeiro, é necessário incluir a biblioteca/classe fstream. Depois, é necessário criar uma instancia da classe, por meio de → ofstream algum_nome; Apos isso é necessario abrir o arquivo, por meio de → algum_nome.open("nome_do_arquivo.extensão"); Depois, é só escrever no arquivo aberto, como → algum_nome << "algum_texto"; E, por fim, fechar o arquivo por meio do método close()

Ler arquivos em C

- Primeiro é necessário incluir a biblioteca stdio.h, depois é necessário criar uma variavel do tipo FILE (struct) *nome_variavel (é como se a variável/struct fosse receber o endereço de memoria do arquivo, no caso, tal endereço sera fornecido pela função fopen), após isso é necessário atribuir a função fopen a nossa variavel, dessa forma → nome_variavel = fopen("nome_do_arquivo.extensão", "como_ele_sera_aberto ('r', 'w')); Depois, é necessario criar um laço de repetição (de preferencia while), deste modo → while((contador = getc(file) != EOF)

Referencias uteis:

- http://linguagemc.com.br/ponteiros-em-c/
- https://www.ime.usp.br/~pf/algoritmos/aulas/footnotes/cast.html
- https://www.ime.usp.br/~pf/algoritmos/aulas/aloca.html
- https://www.guru99.com/c-file-input-output.html
- https://www.tutorialspoint.com/cprogramming/c_file_io.htm
- https://stackoverflow.com/questions/5672746/what-exactly-is-the-file-keyword-in-c