|  |  |  |
| --- | --- | --- |
|  | **PROGRAMAÇÃO DE COMPUTADORES**  PROF. RENATA COSTA | |
|  |  | |
| **Aluno**:   1. Rodrigo de Souza Galvão | | **RGM:**   1. 43679650 |
| **Curso**: Ciência da Computação | | **Semestre/Turno:** 1**º** sem/Matutino |

|  |  |
| --- | --- |
| Tema:  **Linguagem C** | Data:  **02/04/2025** |

**Estrutura Básica C:**

#include <stdio.h>

int main() {

return 0;

}

**#include:** Define a biblioteca padrão da linguagem C, na qual define todos os comandos essenciais da mesma, como printf, scanf, etc.

**Int main():** Define a função principal do programa, tudo que for colocado dentro dessa função, será executado de forma linear (linha por linha, de cima para baixo).

**Return 0:** Define um valor de retorno para ‘dizer’ que o programa foi executado corretamente. Toda **função** precisa ter um valor de retorno, com exceção funções **void**.

**Tipos de variáveis:**

Em C, nós temos tipos de variáveis, que podem ser usadas para armazenar dados temporários (não são salvos no disco rígido), entre os tipos de dados, temos:

|  |  |  |
| --- | --- | --- |
| **Tipos** | **C** | **Explicação** |
| **Inteiro** | Int | Usada para armazenar somente números inteiros (sem casas decimais), positivos e negativos |
| **Real** | Float  Double | Usada para armazenar todo tipo de número, desde raízes não exatas, números com casas decimais, frações, etc. |
| **Caractere** | Char | Usada para armazenar apenas um caractere, que pode ser qualquer símbolo (letras, números, caracteres especiais, e etc. |
| **Texto** | Char[] | Usada para armazenar textos, como por exemplo campos de nomes, descrições, entre outros tipos de rótulo. |

**Declarações de Variáveis:**

Para se declarar variáveis em C, não é um processo muito trabalhoso, basta seguir a seguinte estrutura:

Tipo-da-variavel nome-da-variavel;

ou

Tipo-da-variavel nome-da-variavel = valoratribuido;

Exemplos práticos:

int x = 10; = (Define uma variável de tipo inteiro, com valor inicial 10);

char[50]; = (Define uma string de texto, com espaço para 50 caracteres);

float num1; (Define uma variável de tipo real, com o nome de ‘num1’);

**Atribuição de valores:**

 maior = a;

a variável que está à esquerda, recebe o valor que está na direita. De forma didática:  
maior << a;

**Operadores:**

Em C, nós temos 3 tipos de operadores, sendo eles:

* Relacionais
* Aritméticos
* Lógicos

**Operadores Relacionais:**

São operadores usados em comparações entre dois valores. Veja a tabela:

|  |  |  |
| --- | --- | --- |
| Relacional | | |
| Operador | Descrição | Exemplo |
| == | Igual | a == b |
| >= | Maior ou igual | a >= b |
| <= | Menor ou igual | a <= b |
| < | Menor | a < b |
| > | Maior | a > b |
| != | Diferente | a != b |

**Operadores Aritméticos:**

São operadores usados para fazer cálculos entre dois valores. Veja a tabela:

|  |  |  |
| --- | --- | --- |
| Aritméticos | | |
| Operador | Descrição | Exemplo |
| + | Soma | 5 + 2 = 7 |
| - | Subtração | 3 – 2 = 1 |
| \* | Multiplicação | 5 \* 2 = 10 |
| / | Divisão | 4 / 2 = 2 |
| % | Resto da divisão | 5 % 2 = 1 |

**Operadores Lógicos:**

São operadores usados para estabelecer condições em **Ifs** e **Loops**. Veja a tabela:

|  |  |  |
| --- | --- | --- |
| Lógicos | | |
| Operador | Descrição | Exemplo |
| && | E – Significa que ambas as condições precisam ser verdadeiras para retornar um valor verdadeiro. | (x > 5 && x < 13) |
| || | Ou – Significa que pelo menos umas das condições precisa ser verdadeira para retornar um valor verdadeiro. | (x != 0 || y != 0) |

**Estruturas de comparação lógica (if):**

Estruturas de comparação lógica (if) são usadas para fazer testes lógicos com valores ou variáveis, e fazer uma verificação, em caso de a condição ser verdadeira, acontece um evento, e se a condição for falsa, acontece outro. Veja os exemplos:

    if (b > maior) {maior = b};

nesse exemplo, caso o valor b, seja maior que a variável ‘maior’, essa variável ‘maior’ recebe o valor de b.

if (nf >= 6){

            printf("o Aluno foi aprovado pela AF. Nota final: %d", nf);

        } else {

            printf("O aluno foi reprovado e esta de DP. Nota final: %d\n", nf);

        }

Nesse outro exemplo, é feita uma verificação se a variável nf é maior ou igual a 6. Caso seja, é imprimido que o aluno foi aprovado, **caso contrário (else),** o aluno é reprovado.

**Laços de repetição (for, while, do-while):**

Laços de repetição (loopings) são usados para executar uma determinada função enquanto uma condição for verdadeira.

**While:**

while (a1 < 0 || a1 > 5){

        //esse while garante que não cause looping infinito ao digitar caracteres errados

        while (getchar() != '\n');

        printf("nota A1 invalida, digite a nota correta\n");

        scanf("%d", &a1);

        }

Nesse código, é feita uma verificação se a1 é menor que zero ou maior que 5, se for verdadeiro, o código é executado várias vezes, quando a condição se torna falsa, o código avança e passa do loop.

**Do-While:**

do {

            lastnum = number % 10;

            printf("%d", lastnum);

            number /= 10;

        } while (number != 0);

Tanto o while e o do-while possuem funções semelhantes, mas a principal diferença entre ambos, é que o do-while é executado pelo menos uma vez. Exemplo:

Int i = 5;

Do {

//código de exemplo

} while (i > 9);

Nesse exemplo, i recebe o valor 5, porém, mesmo que esse valor não valide a condição i > 9, o código é executado pelo menos uma vez.

**For:**

Diferente dos anteriores, o for segue uma sintaxe diferente, sendo essa:

for (variável-controle; condição; incremento);

a variável de controle é usada para dizer quantas vezes o for será executado, ou definir um valor inicial.

a condição relaciona a variável de controle com algum outro valor, e o incremento pode ser soma, subtração, etc. Exemplo de código:

for (int i = 3; i <= opt; i++){

        actual = calcFib(starter, next);

        starter = next;

        next = actual;

        printf("%d\n", actual);

    }

É definido a variável de controle como 3 (i = 3). Enquanto o i for menor que o valor de opt, i será acrescido de 1 (i++);

Ou seja, suponha que opt = 10. Com o i iniciando em 3, o código vai executar 7 vezes, até i ser igual ao valor de opt.

**Switch-case:**

o switch-case é usado principalmente para com base em um valor de escolha do usuário, ocorrerá um evento diferente. De forma ilustrada:

um programa tem quatro opções (cases);

o usuário escolhe uma opção;

dependendo da opção, ocorrerá um caso diferente.

No exemplo do código:

switch (opt){

        case '1':

//evento do caso 1

            break;

        case '2':

  //evento do caso 2

            break;

        case '3':

//evento do caso 3

            break;

        case '4':

//evento do caso 4

            break;

        default:

//evento caso a opção escolhida seja inválida.

    }

Sobre o código, são quatro opções (1 à 4), conforme o usuário digitar um desses números, ocorrera um caso diferente, e caso o número da opção digitada seja inválido, ocorre o evento do caso padrão (default).

O ‘**break’,** serve para quebrar a execução do código e impedir que as outras linhas sejam executadas além da escolhida.

**Structs:**

**Funções:**

**Ponteiros:**