Resumo Programação: Teste 02

Um algoritmo de ordenação de números deve realizar os seguintes passos:

1. definir N números, que o usuário deseja que sejam organizados;
2. ordenar tais números, por meio de troca de pares, troca de pares consecutivos, ou outros métodos;
3. Mostrar os resultados na tela.

Para ordenar dados de um arquivo, o programa deve carregar o arquivo selecionado, para então estruturar seus dados, ordená-los, e depois mostrar o resultado para o usuário.

Implementação:

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <conio.h>

int inputDados (int vn []) {

int i, a, n;

printf("\nDigite a quantidade de números: ");

scanf("%d", &n);

for (i = 1; i <= n; i++) {

printf("Digite o valor do número %d: ", i);

scanf("%d", &a);

vn[i-1] = a;

}

return (n);

}

int ordenadorDados (int n, int vn []) {

int troca, a, pv, i;

do {

troca = 0;

pv = vn[0];

for(i=1; i<n; i++){

a = vn[i];

if (pv > a){

vn[i-1] = a;

vn [i] = pv;

troca++;

} else {

pv = a;

}

}

} while (troca);

}

void outputDados (int n, int vn[]) {

int i;

printf("\n--- Todos os %d números do vetor ---\n", n);

for(i=0; i<n; i++){

printf("%d ", vn[i]);

}

printf("\n-------------------------------------\n");

}

int main()

{

setlocale(LC\_ALL, "Portuguese");

int n, vn[100];

n = inputDados(vn);

outputDados(n, vn);

ordenadorDados(n, vn);

outputDados(n, vn);

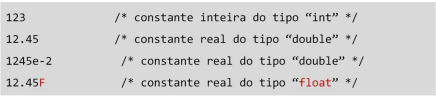
getch();

return 0;

}

Existem vários tipos básicos, também chamados de primitivos, na linguagem C, sendo eles: char, unsigned char, short int, unsigned short int, int, long int, unsigned long int, float e double. Tais tipos possuem valores mínimos e máximos, além de tamanhos que os distinguem, e a compreensão de cada um desses tipos leva a melhor utilização de memória, uma vez que cada variável recebe um tipo que melhor se adequa ao seu valor de uso, durante o programa, evitando que haja uso desnecessário de memória.

Uma constante é armazenada na memória, e sua sintaxe indica um tipo, dos citados anteriormente, vide slide:



Uma variável consiste em um espaço de memória, utilizado para armazenar um dado, e não representa uma variável no sentido matemático. Tal variável possui um tipo e nome, onde o nome identifica o espaço na memória onde ela se encontra, e o tipo indica a sua natureza.

Variáveis devem ser declaradas de maneira explícita, entretanto, podem ser declaradas em conjunto, de forma em que são declaradas várias variáveis do mesmo tipo em uma única linha de código:

int variavel; // declara uma variável do tipo int

float variavel; // declara uma variável do tipo float

int variavel1, variavel2; // declara duas variáveis do tipo int

float variavel1, variavel2; // declara duas variáveis do tipo float

Variáveis só podem armazenar um valor de mesmo tipo que foram declaradas: Uma variável do tipo int, caso seja atribuída o valor 5.2, só irá armazenar o valor 5, pois esta só pode receber números inteiros.

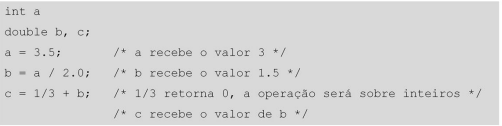
Uma variável pode ser apenas declarada, como visto anteriormente, e posteriormente atribuída a um valor, entretanto, variáveis também podem ser declaradas e inicializadas em uma única linha de código, por exemplo:

int variavel = 1; // a variável é declarada, e recebe o valor 1, na mesma linha.

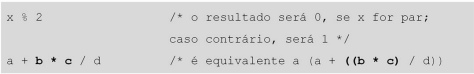
Uma variável só pode ser utilizada caso possua valor já definido anteriormente no código. Caso o programa tente realizar uma operação com uma variável que ainda não foi inicializada, o programa retornará um erro.

Operadores são usados para realizar operações entre variáveis e constantes, podendo serem dos tipos: Aritmético, atribuição, incremento/decremento, relacionais/lógicos, dentre outros.

Em operações aritméticas, o operando menos preciso é convertido para o tipo do operando de maior precisão, e a divisão entre números inteiros descarta a parte fracionária, vide slide:



O operador “%” aplica-se a operações com números inteiros, onde esse operador é utilizado para o retorno da “sobra” da divisão. A ordem para realização das operações segue a regra matemática:



Operadores de Incremento e decremento são utilizados para incrementar ou decrementar de uma unidade o valor de uma variável, podendo ser implementado antes ou depois da variável ser utilizada, e não se aplica a expressões. “a++” Incrementa uma unidade a variável “a”, após sua utilização, “++a” incrementa uma unidade a variável “a”, antes de sua utilização. Na linha de código “b = a++ \* 2”, b recebe o dobro do valor de “a” original, entretanto, a variável “a” não possui seu valor alterado pelo operador de multiplicação, logo seu valor é apenas incrementado em uma unidade.

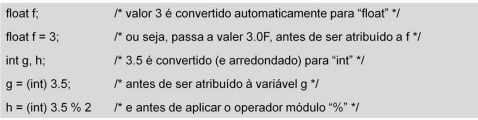
Operadores relacionais comparam dois valores, sejam constantes ou alocados em variáveis, retornando 0 caso a comparação seja falsa, e 1, caso a comparação seja verdadeira.

Para os operadores lógicos, o processo de avaliação das operações é feito da esquerda para a direita, e quando o resultado já pode ser dado desde a primeira operação, a segunda não será analisada.

O Operador sizeof armazena número de bytes ocupados por um tipo, dentro de uma variável:

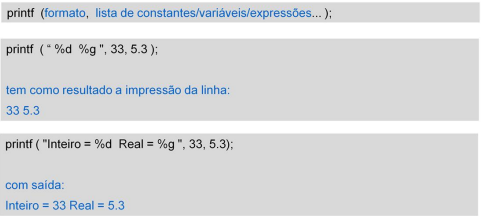
int variavel = sizeof(float); // variavel recebe o valor 4

Os operadores de conversão de tipo podem ser usados de maneira explícita, e a conversão de tipo é automática na avaliação de tal expressão, vide slide:



Entrada & Saída:

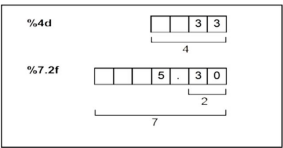
“printf” realiza a saída de um valor, dado seu tipo:



Os tipos de saída são especificados de acordo com os operadores %c, %d, %u, etc, onde cada um representa um tipo.

Os chamados Escape Caracters são utilizados para a formatação do código, como na quebra de linha, comentários do desenvolvedor e etc.

A saída também pode ter tamanho de campo especificado, conforme o slide:



A função “scanf” segue a mesma metodologia da função “printf”, onde é especificado um tipo, para ser lido e armazenado em uma variável.

As tomadas de decisão podem ser realizadas pela função “if”, onde uma expressão é analisada, e caso seja verdadeira (saída 1), o bloco de comandos abaixo é executado, e caso seja falsa (saída 0), outro bloco de comandos é executado. Note que uma função “if” pode ser utilizada dentro de outra função “if” mais exterior, e assim sucessivamente.

Também é possível utilizar o método #define, onde pode-se definir uma função e seus parâmetros, a condição a ser comparada, e a saída, para o caso verdadeiro ou falso, por exemplo:

#define func(a, b)(a>b ? a : b)

A função recebe dois valores, os compara, e caso verdadeira, retorna o valor referente a “a”, e caso falsa, retorna o valor referente a “b”