

ATIVIDADE 2 - ESTRUTURAS DE DADOS - 52_2025**Período:**26/05/2025 08:00 a 06/07/2025 23:59 (Horário de Brasília)**Status:**ABERTO**Nota máxima:**0,50**Gabarito:**Gabarito será liberado no dia 07/07/2025 00:00 (Horário de Brasília)**Nota obtida:****1ª QUESTÃO**

Dentro da lógica de programação, algoritmos e estruturas de dados, um dos recursos mais importantes para o desenvolvimento de aplicações flexíveis, é o conceito de structs. Esse tipo de recurso é utilizado, inclusive, em linguagens orientadas a objetos e, por isso, devem ser dominadas por programadores(as) que têm ciência de sua importância.

OLIVEIRA, Pietro Martins de; PEREIRA, Rogério de Leon. **Estruturas de Dados I**. Maringá: Unicesumar, 2019.

A respeito de estruturas de dados compostas, avalie o trecho de código, em Linguagem C, a seguir:

Linha	Código
	struct notas {
01	float bim1;
02	float bim2;
03	float bim3;
04	float bim4;
05	}
06	struct notas alunos_notas
07	
08	40
	;

Com base no exposto acima, analise as afirmações a seguir:

I – Entre as linhas 01 e 06 temos a definição de uma struct que permite o armazenamento de dados numéricos com casas decimais.

II – Caso o autor do código quisesse, poderia alterar o tipo de dados das linhas 02 a 05 de float para double, sem alterar a semântica do código.

III – Aparentemente, na linha 08, a declaração do vetor de struct intitulado alunos_notas está com erro sintático.

De acordo com as afirmações acima, é possível dizer que está(ão) correta(s) a(s) afirmativa(s):

ALTERNATIVAS

- ☐ I, apenas.
- ☐ II, apenas.
- ☐ III, apenas.
- ☒ I e II, apenas.
- ☐ II e III, apenas.

2ª QUESTÃO

Como podemos acessar uma variável usando seu endereço? Uma vez que os endereços de memória também são apenas números, eles também podem ser atribuídos a alguma outra variável. As variáveis que são usadas para manter endereços de memória são chamadas Ponteiros. Um ponteiro não é, portanto, nada além de uma variável que contém um endereço de alguma outra variável.

Partindo desse entendimento, analise as afirmações a seguir:

- I. Um ponteiro em C é usado para alocar a memória dinamicamente, ou seja, no tempo de execução.
- II. A variável ponteiro pode pertencer a qualquer um dos tipos de dados, como int, float, char, double, short etc.
- III. Uma variável (normal) armazena valor, enquanto a variável ponteiro armazena o endereço da variável.
- IV. O conteúdo do ponteiro sempre é um número inteiro, ou seja, um endereço.

É correto o que se afirma em:

ALTERNATIVAS

- ☐ I e II, apenas.
- ☐ III e IV, apenas.
- ☐ I, II e IV, apenas.
- ☐ II, III e IV, apenas.
- ☒ I, II, III e IV.

3ª QUESTÃO

Pilhas estáticas são estruturas de dados que podem ser implementadas por meio de vetores estáticos. Tais vetores geralmente são alocados de maneira contígua em memória, ou seja, seus elementos ficam dispostos um em seguida do outro, em memória. Além disso, o encapsulamento dos elementos conceituais de uma estrutura de dados pode ser feito com o auxílio de structs.

OLIVEIRA, Pietro Martins de; PEREIRA, Rogério de Leon. **Estruturas de Dados I**. Maringá: Unicesumar, 2019.

Dessa forma, observe o trecho de código abaixo, no qual implementamos parcialmente uma pilha na qual os elementos serão inseridos e removidos apenas por uma de suas extremidades.

Linha	Código
	#include <stdio.h>
01	
02	#define tamanho 10
03	
04	struct tpilha {
05	int dados
06	<i>tamanho</i>
07	
08	};
09	int topo;
10	};
11	
12	struct tpilha pilha;
13	
14	void pilha_mostrar() {
15	int i;
16	printf(" "
17	");for(i = 0; i < pilha.topo; i + +)printf(" printf("
18	
19	\\n\\n");
	}

Com base no exposto, avalie as afirmações a seguir:

I – Da forma como o código foi escrito, podemos dizer que nossa pilha pode armazenar, no máximo, dez elementos.

II – Se a pilha estiver cheia, podemos dizer que o laço da linha 15 deveria realizar 10 iterações.

III – Normalmente, em uma pilha, os dados são inseridos e também removidos a partir do topo.

De acordo com as afirmações acima, é possível dizer que está(ão) correta(s) a(s) afirmativa(s):

ALTERNATIVAS

- ☐ I, apenas.
- ☐ I e II, apenas.
- ☐ I e III, apenas.
- ☐ II e III, apenas.
- ☒ I, II e III.

4ª QUESTÃO

Ponteiros desempenham um papel crucial quando uma variável precisa ser acessada em diferentes partes de um programa. Nesse contexto, é comum encontrar diversos ponteiros distribuídos por várias seções do código, cada um apontando para a variável que contém os dados necessários. Uma vantagem significativa dessa abordagem é que, se esses dados forem alterados, não há preocupação, pois todos os ponteiros no programa estão direcionados para o endereço onde os dados atualizados residem. Essa flexibilidade oferecida pelos ponteiros é fundamental para garantir a eficiência e a consistência na manipulação de dados em diferentes partes do código.

Considerando o uso de ponteiros em estruturas de dados, analise o seguinte código em C:

```
#include <stdio.h>

int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    int *ptr = arr;
    printf("%d\n", *ptr + 2);
    printf("%d\n", *(ptr + 2));
    printf("%d\n", ptr[2]);
    printf("%d\n", *ptr++);
    printf("%d\n", (*ptr)++);
    return 0;
}
```

Fonte: Elaborado pelo professor, 2024.

Dado o contexto e analisando o código, qual será a saída impressa ao executar este código?

ALTERNATIVAS

- ☒ 3, 3, 3, 1, 2
- ☐ 3, 3, 3, 1, 3
- ☐ 3, 5, 3, 1, 2
- ☐ 3, 5, 3, 2, 2
- ☐ 3, 5, 3, 2, 3

5ª QUESTÃO

Um dos temas que diferenciam a linguagem C das demais linguagens é a possibilidade de podermos manipular a memória de um dispositivo computacional através dos ponteiros. Ponteiros são um recurso muito útil, especialmente no que tange um maior controle sobre os endereços de memória nos quais os conjuntos de dados de um programa se encontram armazenados em uma aplicação.

OLIVEIRA, Pietro Martins de; PEREIRA, Rogério de Leon. **Estruturas de Dados I**. Maringá: Unicesumar, 2019.

Com isso em mente, observe o código-fonte, em linguagem C, a seguir:

Linha	Código
01	#include <stdio.h>
02	#include <stdlib.h>
03	
04	int xi;
05	int *ptr_xi;
06	
07	float xf;
08	float *ptr_xf;
09	
10	int main() {
11	int xi;
12	int *ptr_xi = xi;
13	system("pause");
14	return(0);
15	}

Com base em seus conhecimentos sobre ponteiros, bem como no código acima, avalie as afirmações a seguir:

I – Pode-se dizer que a declaração das variáveis das linhas 07 e 08 são desnecessárias, pois não estão sendo utilizadas.

II – O código não irá compilar, pois na linha de código 12 temos uma atribuição sintaticamente inválida.

III – Mesmo um cast, seria impossível tentar fazer com que ptr_xf recebesse o endereço de xi.

De acordo com as afirmações acima, é possível dizer que está(ão) correta(s) a(s) afirmativa(s):

ALTERNATIVAS

- ☒ I, apenas.
- ☐ II, apenas.
- ☐ III, apenas.
- ☐ I e II, apenas.
- ☐ II e III, apenas.

6ª QUESTÃO

Os ponteiros são elementos fundamentais na linguagem C, conferindo-lhe uma notável flexibilidade e poder. Eles funcionam como variáveis especiais que armazenam endereços de memória de outras variáveis, permitindo acessá-las diretamente. Quando dizemos que um ponteiro "aponta" para uma variável, significa que ele detém o endereço dessa variável na memória. Essa capacidade de apontar para diferentes tipos de variáveis, como inteiros, ponto flutuante, duplos, entre outros, confere aos ponteiros uma versatilidade excepcional.

Fonte: Elaborado pelo professor, 2024.

Dado o contexto, qual a função do operador de referência (&) em estruturas de dados utilizando ponteiros em linguagem C?

ALTERNATIVAS

- ☒ Retorna o endereço de memória da variável.
- ☐ Aloca dinamicamente memória para a variável.
- ☐ Libera a memória alocada dinamicamente pelo ponteiro.
- ☐ Retorna o valor contido na posição de memória apontada pelo ponteiro.
- ☐ Desreferencia o ponteiro, acessando o valor armazenado na posição de memória apontada.

7ª QUESTÃO

Os conceitos de variáveis homogêneas e heterogêneas, vetores, matrizes e registros, listas, filas, etc, são os elementos básicos necessários para um processo lógico e estruturado de formação do conhecimento, também para construir estruturas de dados mais avançadas.

Fonte: Elaborado pelo professor, 2024.

Dado o contexto, qual a estrutura de dados é a mais adequada para armazenar informações de alunos, contendo nome, idade, sexo, curso e média final?

ALTERNATIVAS

- ☐ Fila
- ☐ Pilha
- ☒ Registro
- ☐ Árvore binária
- ☐ Lista encadeada

8ª QUESTÃO

Listas, pilhas e filas são estruturas de dados, que nos permitem organizar e interagir com nossos dados e organizá-los de diferentes formas. Em C++ existem bibliotecas que nos permitem implementar esses tipos de estruturas, o que torna nosso trabalho mais simples do que se tivéssemos de criá-las manualmente.

GOULART, Gabriel. **Listas, pilhas e filas em C++**. Portal GSTI. Disponível em: <https://www.portalgsti.com.br/2018/04/listas-pilhas-e-filas-em-c.html> acesso em: 09/02/2022.

A partir dessas informações, avalie as asserções a seguir e a relação proposta entre elas:

I. Pilha é uma coleção de elementos, que segue a ordem LIFO. O que significa que o elemento que é inserido mais recentemente será removido primeiro. Uma pilha tem uma restrição de que a inserção e exclusão do elemento só pode ser feita a partir de apenas uma extremidade da pilha e chamamos essa posição de topo.

PORQUE

II. Fila é uma estrutura de dados que segue o princípio FIFO. Sendo que o elemento adicionado primeiro na fila será o que será removido primeiro. Os elementos são sempre adicionados na parte de trás e removidos da frente.

A respeito dessas asserções, assinale a opção correta.

ALTERNATIVAS

- ☐ As asserções I e II são proposições verdadeiras e a II é uma justificativa correta da I.
- ☒ As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa correta da I.
- ☐ A asserção I é uma proposição verdadeira e a II é uma proposição falsa.
- ☐ A asserção I é uma proposição falsa e a II é uma proposição verdadeira.
- ☐ As asserções I e II são proposições falsas.

9ª QUESTÃO

É possível fazer alocação dinâmica na memória por meio da função MALLOC. A função malloc(), acrônimo para Memory Allocation, é uma função da biblioteca stdlib.h que recebe como argumento números inteiros positivos (size_t), que irão representar o número de bytes que desejamos alocar. Essa função retorna um ponteiro contendo o endereço do bloco alocado ou NULL em caso de falha.

Disponível em: <<https://www.cprogressivo.net/2013/04/Como-usar-a-funcao-malloc-para-alocar-memoria-em-linguagem-C.html>>, Acessado em 26.mar.2019.

Assinale a alternativa correta que mostra a sintaxe na linguagem C para alocar um espaço na memória para um inteiro usando esse comando.

ALTERNATIVAS

- ☐ ptr = malloc(int)
- ☐ ptr = (int)malloc
- ☐ ptr = (int *) malloc(int)
- ☐ ptr = malloc(sizeof (int))
- ☒ ptr = (int *) malloc(sizeof (int))

10ª QUESTÃO

No trecho de código a seguir, foi iniciado o desenvolvimento de uma fila estática que é implementada com auxílio de uma struct contendo um campo "dados" (vetor), um campo "ini" que deverá armazenar o índice do primeiro elemento da fila e um campo "fim" que deverá armazenar o índice do último elemento da fila.

OLIVEIRA, Pietro Martins de; PEREIRA, Rogério de Leon. **Estruturas de Dados I**. Maringá: Unicesumar, 2019.

Observe:

Linha	Código
	#define tamanho 5
	struct tfila {
	int dados
	<i>tamanho</i>
01	};
02	int ini;
03	int fim;
04	};
05	struct tfila fila;
06	
07	void remove() {
08	int i;
09	for (i = 0; i < tamanho; i++) {
10	fila.dados
11	<i>i</i>
12	<i>i</i>
13	= fila.dados
14	<i>i + 1</i>
15	<i>i + 1</i>
16	<i>i + 1</i>
17	};
18	}
	fila.dados
	<i>fila.fim</i>
	= 0;
	fila.fim--;
	}

Com base em seus conhecimentos sobre filas estáticas, tomando como referência o código-fonte acima, avalie:

- I – A função definida entre as linhas 11 e 18 é do tipo void, por isso não retorna valor algum.
- II – A função remove() realiza o deslocamento dos dados da fila em uma posição da direção do fim à direção do início da fila.
- III – Podemos considerar que o início da fila será, nesse caso, a posição 0 do vetor de dados da struct tfila.

De acordo com as afirmações acima, é possível dizer que está(ão) correta(s) a(s) afirmativa(s):

ALTERNATIVAS

- ☐ I, apenas.
 - ☐ I e II, apenas.
 - ☐ I e III, apenas.
 - ☐ II e III, apenas.
 - ☒ I, II e III.
-