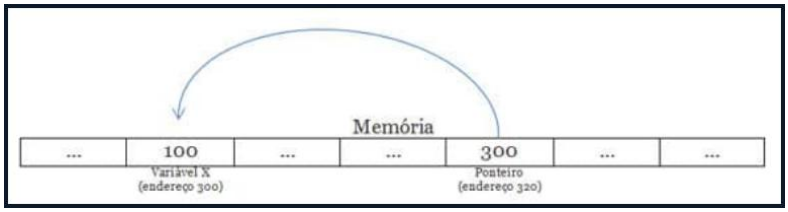
A memória é um componente do computador responsável pelo armazenamento de dados e instruções. Ela é composta por palavras, sendo cada palavra identificada unicamente a partir de um endereço, ou seja, um endereço de memória. A alocação de memória na linguagem C pode ser de três tipos:

Estática, automática e dinâmica

Sabemos que todo dado armazenado em memória possui um endereço e que a definição de um ponteiro é uma variável que guarda o endereço de memória, ou seja, a localização do dado. Diante disto, a figura apresenta uma memória que armazena a variável “x” e o ponteiro para a variável “x”. Como o conteúdo do ponteiro é um endereço, a seta indica a relação entre o ponteiro e a variável que ele aponta. Qual o valor da variável “x” e este dado está armazenado em qual endereço de memória, isto é, qual é o valor armazenado pelo ponteiro “pt_x”?



x = 100; pt_x = 300

Estudamos sobre os tipos de estrutura de dados. A estrutura de dados em que o primeiro elemento inserido é o primeiro elemento a ser retirado é denominada:

Fila

As estruturas de dados heterogêneas são conjuntos de dados formados por tipos de dados diferentes. Sendo assim, selecione uma das principais estruturas de dados heterogêneas.

Registro

Para acessar os membros de uma estrutura de dados struct, são empregados dois tipos de operadores: Operador de membro de estrutura . (operador de ponto) e Operador de ponteiro de estrutura -> (operador de seta). Selecione a opção correta que apresenta o comando para acessar o elemento rua de uma struct endereço que foi referenciada diretamente pela variável x.

Printf(“%s”, x.rua);

Considere que você está desenvolvendo um novo software para catalogar os jogadores de futebol. Você modelou o seu programa utilizando as structs apresentadas abaixo.

```
typedf struct time {
    char nome[50];
} Time;

struct Jogador {
    int nr_camisa;
    char nome[30];
    Time t;
} jogador;
```

Considerando que você precise cadastrar um novo jogador chamado Manoel da Silva, que irá atuar no time da Estudante Atlético Club, com a camisa número 10, qual alternativa apresenta a forma correta de inserir os dados?

jogador.camisa = 10;

strcpy(jogador.nome, “Manoel da Silva”);
strcpy(jogador.t.nome, “Estudante Atlético Club”);

Considere o trecho de código abaixo:

```
typedef struct {
    char nome[200];
    int idade;
    float salario;
} Funcionario;
Funcionario func[10];.
```

Qual das opções apresenta a sintaxe correta na qual seja preenchido corretamente o elemento de uma posição do vetor de struct?

&func[índice].idade;

Estudamos sobre modularização na programação, que é a divisão do código do programa em sub-rotinas. Assinale a alternativa que contenha tipos de sub-rotinas:

Procedimento e função

Assinale a alternativa que contenha a afirmativa correta sobre função e procedimento:

As funções sempre retornam algum valor a quem as chamou, e os procedimentos não retornam valor algum.

Na linguagem de programação C, as funções predefinidas estão nas bibliotecas da linguagem. Cada biblioteca padrão tem um cabeçalho que contém os protótipos para todas as funções, assim como definições de vários tipos de dados e constantes que são necessárias para as mesmas. Uma dessas bibliotecas tem a seguinte explicação: Funções de entrada e saída padrão. Assinale-a:

Stdio.h

Considere o Programa C, listado a seguir.

```
#include <stdio.h>
int main(void) {
      int a, *y, z;
      x = 5;
      y = &x;
      z = 10;
      scanf(“%d”, y);
      printf(“%d %d ”, x, z);
}
```

Assinale a alternativa que representa o que será impresso pelo programa se o usuário digitar 15, como entrada de dados:

15 10

Na programação em módulos, o programa principal se comunica com as sub-rotinas através de passagem de parâmetro. Sobre os tipos de passagem, assinale a sentença correta:

Os parâmetros atuais ou reais são os argumentos da função.

Analise o seguinte código implementado na linguagem C com passagem de parâmetro por referência entre a função main e a função soma.

```
int soma(int *a, int *b) {
       *a = *a + *b;
       return *a;
}
int main() {
       int x=5, y=3;
       y = soma(x, y);
       printf("%d", x+y);
       return(0);
}
```

Linha 7 y = soma(&x, &y);

Nas linguagens de programação, as variáveis são vinculadas aos seus valores em tempo de execução. E o escopo de uma vinculação é o conjunto de trechos de um programa que se consegue usar as variáveis. No caso da linguagem C, as variáveis são divididas quanto ao escopo em três tipos. Assinale a opção que apresenta esses três tipos:

Variáveis locais, variáveis globais e parâmetros formais

Analise o seguinte código implementado na linguagem C.

```
int soma(int a, int b) {
       int s;
       s = a + b;
       return s;
}
int main() {
       int x=5, y=10;
       int z ;
       z = soma(x, y);
       printf("%d", z);
       return(0);
}
```

Assinale a opção correta:

O escopo de vinculação da variável s está definido pela função soma.

Listas lineares ordenadas são estruturas de dados nas quais a posição relativa dos seus elementos reflete uma ordem que sempre deve ser respeitada. Suponha um vetor V em linguagem C com n posições que implementa uma lista desse tipo. É correto afirmar-se que:

V [1] > V [0] , então, V [i + 1] > V [i]

Sobre listas lineares alocadas sequencialmente, é verdadeiro que:

São eficientes no acesso ao elemento da lista.

Sobre listas lineares implementadas por meio de alocação encadeada, são feitas as seguintes afirmativas:
I) Em uma lista ordenada, os endereços de memória de seus nós também estarão ordenados.
II) Uma lista duplamente encadeada não pode ser usada para implementar uma lista ordenada.
III) Para acessar um nó, é necessário percorrer todos os seus predecessores.
A alternativa que contém apenas afirmativa(s) verdadeira(s) é:

Somente a III está correta.

"A alocação sequencial aloca os espaços de memória em tempo de execução, não sendo seus endereços conhecidos previamente. Assim, para acessar um nó, é necessário percorrer os antecessores, pois cada um guarda o endereço do seguinte. Pela mesma razão, não é possível se afirmar qual a relação entre os endereços dos nós. Por fim, o duplo encadeamento não viola a ordem existente entre os elementos de uma lista ordenada."

A concatenação de duas listas A e B é uma operação que faz o último nó de A ser seguido pelo primeiro nó de B, unindo-as e produzindo uma lista cujo número de nós é igual à soma do número de nós de cada lista individual. Considere as listas não ordenadas L1 e L2, ambas implementadas por alocação simplesmente encadeada. L1 possui m nós e L2 possui n nós. Sobre a concatenação de L1 e L2, é correto afirmar-se que:

Serão percorridos *m* nós

"Concatenar L1 e L2 significa fazer o último nó de L1 apontar para o primeiro de L2. Como em alocação encadeada não é possível o acesso direto, será necessário percorrer todos os nós de L1 para atingir seu último nó e, assim, realizar o apontamento para o primeiro nó de L2. Entretanto, L2 não precisa ser percorrida. Assim, serão percorridos *m* nós."

Considere uma pilha que inicialmente possui os seguintes elementos: A, D, R, K, P. A execução de uma operação de desempilhamento (pop) retira o elemento "A". Em seguida são executadas as operações pop e push (empilha o elemento passado como parâmetro) na sequência: pop (), push (H), push (A). Após essas operações, é correto afirmar que a pilha resultante é:

A, H, R, K, P

"Uma pilha somente admite remoção por uma extremidade, chamada topo. Se a execução de pop () removeu A, então o topo da pilha está à esquerda. Assim, repetir pop () removerá D, e push (H), push (A) empilharão H e A nessa ordem, de forma que A está no topo."

Qual das afirmativas apresenta corretamente uma vantagem de se implementar pilhas em alocação encadeada?

Evita o desperdício de memória.

"Na alocação encadeada, a memória é alocada em tempo de execução. Isso permite que sejam alocadas tantas porções de memória quanto se necessite, o que faz com que não haja um limite teórico. O limite é, de fato, a memória disponível no computador."

A regra de que o elemento mais antigo numa lista é o primeiro a ser removido corresponde, respectivamente, ao conceito e ao nome de:

FIFO e FILA

"O elemento mais antigo foi o primeiro a ser inserido e é o primeiro a ser removido, isso corresponde ao conceito de First in, First Out (FIFO) e caracteriza a FILA."

Uma empresa desenvolveu um teclado sem fio para ser utilizado em celulares. Entretanto, devido a vários fatores, a transmissão dos eventos (digitação) para o celular é mais lenta do que a velocidade de digitação de alguém experiente. Para evitar a perda de dados, a empresa resolveu introduzir um buffer para guardar as teclas

digitadas e enviar para o celular à medida que este for processando os dados. A estrutura de dados adequada para implementar esse buffer é uma:

Fila

"O buffer tem que garantir que a ordem dos caracteres digitados não se altere. A estrutura de dados que garante isso é a fila."

Analise as afirmativas abaixo e marque a opção correta.
1 – Não existe algoritmo de ordenação capaz de executar em um tempo inferior à complexidade de O (n log n).
2 – Métodos de ordenação interna executam em memória principal (RAM) somente, enquanto métodos de ordenação externa executam em memória secundária, porém podem usar memória principal também.
3 – Somente algoritmos de ordenação não baseados em comparação podem executar em um tempo inferior à O(n log n).

As afirmativas 2 e 3 estão corretas.

"A afirmativa 1 está incorreta, algoritmos não baseados em comparação entre elementos da sequência a ser ordenada não estão sujeitos ao limite de complexidade de O(*n log n*)."

Marque a alternativa correta em relação à estabilidade de um algoritmo de ordenação.

Um algoritmo estável pode necessitar de menos operações que um algoritmo instável, entretanto, isto não reduz a complexidade computacional do algoritmo.

"A estabilidade pode reduzir o número de operações em algumas instâncias, porém não reduz a complexidade computacional do algoritmo."

Quando apresentamos uma sequência já ordenada para os algoritmos da bolha e para o Insert Sort, é correto afirmar que:

Ambos executam sempre em tempo linear para as instâncias já ordenadas.

"O método da bolha e o Insert Sort executam em tempo linear para o melhor caso. Em ambos algoritmos, o melhor caso é a sequência ordenada."

Comparando o Selection Sort com o Bubble Sort e o Insert Sort, é correto afirmar que:

Os três têm a mesma complexidade computacional.

"Não há base, além da complexidade computacional, para afirmar que um algoritmo é melhor que outro. Como os três têm a mesma complexidade, são considerados equivalentes."

Sobre a estrutura de dados chamada árvore, está correto o que se afirma em:

O maior nível de uma árvore é numericamente igual à sua altura.

"Tanto o nível quanto a altura são calculados contando-se a partir do valor 1, a altura. E a altura da árvore é definida como sendo o maior valor entre os níveis de seus nós."

Sejam as afirmações a seguir:

I. Se duas árvores puderem ser tornadas coincidentes com a permutação de um nó pai com seu nó filho, então essas árvores são isomorfas.
II. Isomorfismo é uma propriedade que não se aplica a árvores representadas por diagramas de inclusão.
III. Duas árvores isomorfas têm a mesma altura.

São verdadeiras, apenas:

A alternativa III

"O isomorfismo entre duas árvores diz respeito à possibilidade de elas serem tornadas coincidentes pela permutação de suas subárvores. Permutar as subárvores apenas alterna suas posições relativas, mas não permite alterar a altura de uma árvore. Isto pode ser provado por contradição. Suponha que a permutação tenha alterado a altura da árvore para torná-la coincidente com a outra. Neste caso, se a altura mudou, obrigatoriamente algum nó mudou de nível, pois não há inserção ou remoção de nó. Então, para que ambas as árvores sejam coincidentes, fez-se necessário permutar as subárvores e mudar o nível de algum nó, o que contraria a definição de isomorfismo."

Árvores binárias são um tipo particular de árvores em que se admite que seus nós possuam até 2 filhos. Sobre tal característica, qual afirmativa apresenta-se correta?

Cada nível tem condição de acomodar o dobro de nós do nível anterior.

"Uma árvore binária, por definição, é uma árvore cujo nó, seja qual for, pode ter até 2 filhos. Assim, o nó raiz pode ter 2 filhos, logo, o nível 2 acomoda o dobro de nós do nível 1. Cada nó do nível 2, por sua vez, pode ter dois filhos. Então, o nível 3 pode acomodar 4 nós. É fácil ver que cada nó de um nível permite acomodar 2 nós filhos no nível seguinte. Logo, cada nível de uma árvore binária permite acomodar o dobro de nós do nível anterior."

Percorrer uma árvore binária é uma forma de realizar sistematicamente uma operação sobre seus nós. Durante o percurso, pode ser necessário acessar um nó mais de uma vez. Sobre este tema, a única opção que apresenta uma afirmativa correta é:

O percurso em ordem simétrica visita todos os nós da árvore somente uma vez.

"Todas as 3 formas de percorrer uma árvore, pré-ordem, ordem simétrica e pós-ordem visitam todos os nós da árvore binária uma única vez. A diferença dos percursos é a ordem com que os nós são visitados."

Árvores binárias de busca (ABB) são uma aplicação de árvores binárias para solucionar problemas de busca. Para isso, uma ABB tem seus nós rotulados de forma que estes guardem uma relação de precedência entre os que pertencem às subárvores esquerda e direita. Marque a única opção que contém uma afirmação correta sobre ABB.

Todos os nós da subárvore esquerda sempre têm rótulos menores que os nós da subárvore direita.

"Uma ABB é definida de maneira que qualquer que seja o nó V da árvore, os nós pertencentes à subárvore esquerda de V têm rótulo menor do que V e os nós pertencentes à subárvore à direita de V têm rótulo maior do que V".

Sobre árvores AVL, analise as afirmações a seguir:

I) Toda árvore cheia é uma árvore AVL.
II) Toda árvore AVL é uma árvore cheia.
III) Balancear uma árvore AVL desregulada implica transformá-la numa árvore completa.

São verdadeiras apenas:

Apenas a alternativa I

"Uma árvore cheia só admite nós com subárvore vazia no último nível. Então, todos os nós dos níveis anteriores não possuem árvores vazias. Pela definição, também inferimos que todos os nós do último nível são folhas, pois, do contrário, existiria pelo menos um nó no penúltimo nível com uma subárvore vazia. Nesta configuração, as subárvores esquerda e direita de um nó v qualquer sempre têm a mesma altura e, portanto, uma árvore cheia é AVL."