1. Explique o que é um ponteiro em C e qual a vantagem em se trabalhar com eles.

R. Ponteiro é uma variável que tem a capacidade de link endereços na memória de outras variáveis. Com Ponteiros podemos acessar o conteúdo de outras variáveis sem precisar altera-la diretamente, por exemplo, digamos um caso em que não conseguimos alterar ou acessar uma determinada variável diretamente, com ponteiros é possível, é justamente nesses casos que devemos a vantagem de utilizar ponteiros.

2. Explique com se declara um ponteiro em C. Cite exemplos.

R. A Declaração de ponteiros é semelhante a variáveis comuns, só que com um asterisco antes do nome da variável.

Ex:. Tipo *nome_da_variavel;

int *circuferencia;

De acordo com o exemplo acima, declarei um ponteiro do tipo inteiro, o (*) que é o que faz dele um ponteiro seguido do nome da variável.

3. Explique a diferença entre o operador de endereço '&' e o operador de conteúdo '*' em C.

R. O operador de endereço (&) ele linca outros endereços de outras variáveis no ponteiro, assim o ponteiro vai apontar pra tal variável.

Ex:. int a, *p;

p = &a;

Já o operador de conteúdo (*) na declaração de variável ele está declarando que aquela variável é um ponteiro, no código, ele possui a função de acessar o conteúdo de tal variável.

Ex:. int a, b, *p;

p = &a;

b = *p;

4. Explique como se acessa o conteúdo apontado por um ponteiro em C e por um ponteiro de ponteiro.

R. Quando declarado e apontado para outra variável, é possível acessar o conteúdo do ponteiro na função printf, basta colocar um asterisco antes do nome da variável.

Ex:. int a, *p;

p = &a;

printf(“%d”, *p);

Já um ponteiro de ponteiro, é um ponteiro que aponta para outro ponteiro, ou seja, um ponteiro que aponta para um endereço de memória que contém outro endereço de memória.

Ex:. int a, *p, **pp;

p = &a;

pp = &p;

printf("%d", **pp);

No exemplo acima, conseguimos alterar o valor da variável a, acessando o conteúdo do conteúdo de pp.

5. Explique como se passa um ponteiro como parâmetro para uma função em C. Cite exemplos.

R. O processo é semelhante à passagem de uma variável comum como parâmetro, só com algumas diferenças, a primeira é na assinatura (cabeçalho) da função, precisa indicar a variável com um asterisco (*) à esquerda do nome da variável. Já a segunda é quando chamamos a função na main. Iremos passar um endereço (&) da variável para o ponteiro, não seu conteúdo. O operador de endereço (&) fica à esquerda do nome da variável.

tipo_da_funcao nome_da_funcao(tipo_do_parametro *nome_do_ponteiro);

nome_da_funcao(&nome_da_variavel);

Ex:. void imprimi(int *n)

...

int a;

imprimi(&a);

...

6. Explique as diferenças entre ponteiros constantes e ponteiros não constantes em C. Cite exemplos.

R. Nos Ponteiros constantes, o ponteiro aponta para um local fixo na memória, e o valor nesse local pode ser alterado porque é uma varável, mas o ponteiro sempre apontará para o mesmo local porque se torna constante. Já em ponteiros não constantes são ponteiros normais, porque além de aponta para uma variável, ele pode posteriormente no programa apontar para outro endereço de memória. Para declarar um ponteiro constante, é semelhante ao declarar uma variável constante normal da forma const. Primeiro definimos o tipo do ponteiro logo o asterisco seguido do 'const' e o nome do ponteiro.

```
tipo_do_ponteiro *const nome_do_ponteiro;
```

Ex:.
```
int a = 16, b = 20;

int *const p = &a;


printf("Ponteiro P apontando para A %d", *p);


p = &b;

printf("Ponteiro P apontando para B %d", *p);
```

Com um ponteiro constante declarando e apontado para um endereço de uma variável, não é possível aponta-lo para outro, o compilador apresenta um erro.

7. Sabemos que podemos declarar ponteiros para int, float, char e demais tipos existentes na linguagem C. Além disso, podemos também declarar ponteiros para funções. Explique a vantagem em manipular um ponteiro para funções. Cite exemplos.

R. Ponteiros de função são usados para apontar para uma determinada função, possibilitando chama-la diretamente, seja como parâmetro ou inicializada. Para declarar um ponteiro de função, primeiro devemos especificar o tipo de retorno da função, logo o nome do ponteiro, à esquerda o asterisco (*), tudo entre parênteses, seguido dos tipos de parâmetros em parêntese.

Ex:.
```
float (*p)( float , flaot );

...

float media( float (*p)( float n1, float n2)){

return (*p) / 2;

}

...


Ou


float calcula(float n1, float n2){

return n1 + n2 / 2;

}

Int main(){
```

```
loat (*num)(float, float);
num = calcula;
printf(“%d”, num(6, 2));
return 0;
}
```