Lista de Revisão de C

1. Responda:
   a) Qual a relação entre escopo e durabilidade de variáveis ? Há como alterar a durabilidade de uma variável independentemente do seu escopo ?
   b) Fale sobre as diretivas de pré-processamento em C. Forneça exemplos de macro-funções, o que as diferenciam das funções convencionais ?
   c) O que é um *typedef* ? exemplifique.
   d) Forneça exemplos de utilização das funções de alocação / liberação de memória (*malloc*, *calloc*, *realloc*, *free*).
   e) Por que é importante iniciar um ponteiro antes do seu uso ? Se um ponteiro armazena um endereço de memória (numero inteiro) por que é necessário declarar um tipo para um ponteiro ?
   f) É possível passar parâmetros para a função *main( )* ? Exemplifique.
   g) Se arquivos armazenam bits, qual a diferença entre arquivos modo texto e arquivos modo binário ?

2. Suponha a alocação de memória para variáveis inteiras, conforme tabela abaixo:

X	Y	Z	
1000	1002	1004	Posição de memória
2	3	7	Conteúdo

   Além disso suponha a existência dos seguintes ponteiros do tipo inteiro:

P1	P2	Ponteiros
5400	5500	Posição de memória
....	.....	Conteúdo armazenado

   a)Escreva os comandos necessários para direcionar os ponteiros p1 para x e p2 para y;
   b) Quais as conseqüências dos comandos abaixo ?

```
I.    printf (“%i”,++*p1);
II.   z = *(--p2);
III.  *p2 = *p2 + 10;
IV.    printf (“%X”, p1);
V.    p2 = &y;
VI.   printf (“%i”,*p2+2);
VII.  printf (“%X”,p2);
```

3. O que são “ponteiros para ponteiros” ? Declare um ponteiro para ponteiro chamado **pp** e faça-o “apontar” para o ponteiro p1 na questão anterior, após isso responda o que será exibido e o que tal valor significa:

a) printf (“%X”, p1);	c) printf(“%i”, *p1):	e) printf(“%i”, **pp):	g) printf (“%X”, &pp);
b) printf (“%X”, pp);	d) printf(“%i”, *pp):	f) printf (“%X”, &p1);	

4. Dada a declaração abaixo:

```
int v[ ]= { 1,2,3,4,5,6,7}, L, *p;
p = v;  /* ou p = &v[0]; */
```

   Descreva o resultado exibido pela execução dos códigos a seguir:

```
for (L=0; L < 7; L++)
   printf("%i\n", *(p+L));
```

```
for (L=0; L < 7; L++)
{  p = p + L;
   printf("%i\n", *p);
 }
```

```
While(p)
  printf(“%i”,*p++);
```

5. Como é declarado um ponteiro tipo void ? Declare um ponteiro do tipo void chamado **pv** e faça-o "apontar" para a variável **x** na questão 2, então multiplique essa variável por dois e após isso incremente essa variável, através de **pv**.
6. Declare um ponteiro para ponteiro, faça-o apontar para *pv* na questão anterior, após isso exiba o valor apontado por *pv* através do ponteiro para ponteiro.

7. Em que consiste uma definição de uma estrutura ? Como é feita a declaração de variável estrutura ? Quais as maneiras possíveis de se ter acesso aos campos de uma estrutura ? Exemplifique.

8. Quais as diferenças e semelhanças entre estruturas e uniões ? A inicialização abaixo está correta ? Justifique sua resposta.

```
union teste{   int numero;
               float real;
               struct teste1{ char nome[M];
                              char  endereco[N];
                            };
       };
```

```
void main(void)
{   union teste data = {20,1.345, {"Joao", "rua x"}};
  .....
}
```

9. As variáveis declaradas como parâmetros de uma função fazem parte do escopo local da função ou do código chamador da função ?

10. O que é "passagem de parâmetros por valor" e "passagem de parâmetros por referência" ? Quais as diferenças entre estes estilos de passagem de parâmetros e suas conseqüências ? Exemplifique.

11. Descreva sintaticamente o seguinte protótipo de uma função para criação de *threads* no padrão POSIX. *int pthread_create(pthread_t * thread, const pthread_attr_t * attr, void * (* start)(void *), void *arg)*

12. Construa a função *void media(int val1, int val2, int val3,.....);* que calcula a média de três valores inteiros. A impressão do resultado deverá ocorrer em comandos no corpo da função *main( )*. Acrescente parâmetros à função para viabilizar isso.

13. Dada a estrutura:

```
typedef struct {   char nome[M];
                   float peso;
                   char sexo;
               }reg;
```

Sem utilizar variáveis globais, pede-se:
   a) Crie um vetor com TAM posições do tipo *reg*.
   b) Crie uma função para entrada de dados utilizando *scanf( )*, acesse os dados através de apontador.
   c) Crie uma função que exibe todo o conteúdo do vetor.
   d) Crie uma função que forneça quantas pessoas do sexo masculino e quantas do sexo feminino foram cadastradas. O protótipo será *void calcula(reg *vet, ...);* e a exibição dos resultados gerados será feita por comandos no *main( )* após a chamada da função.

20. Escreva um programa principal para utilizar as funções abaixo:
   a) Função que converte um número inteiro (unsigned int – numeros entre 00000 e 65535) para string (para tanto não devem ser usadas a função *itoa( )* do C). O protótipo deve ser: *void val_to_string(n, char *s)*. Os operadores: módulo '%' (para obter o resto da divisão inteira) e '/' (para obter o quociente da divisão inteira) podem ser úteis.
   b) Escreva uma função que testa se uma palavra é palíndrome. Exemplos: casacoocasac, omo, casasac

21. Implemente uma função que calcula o número de blocos escritos em um arquivo binário qualquer, esse valor deve ser exibido através de comandos na função *main( )*. Não use ftell( ). ATENÇÃO ! Use o seguinte protótipo para a função *void num_blocos_arq(parâmetros)*.

22. Digite, execute e discuta os seguintes códigos:

a)
```
int vet[ ] = {1,2,3,4,5,6,7,8,9,10}, i , *p
, tam_vet;
p = vet;
tam_vet = sizeof(vet)/sizeof(int);
clrscr();
for (i=0; i<tam_vet; i++,p++)
     printf("%i \n", *p);
getch();
```

b)
```
char str[ ]= "linguagem C", *px, *py,
aux;
px = py = str;
for(;*py != '\0'; py++);
    py- -;
while(px<py)
              { aux = *px;
                *px = *py;
                *py = aux;
                px++;
                py- -;
              }
clrscr( );
puts(str);
getch();
```

c)
```
char *pstr, dia[ ] = {"os dias da semana
são: Seg, Ter, Quar, Qui, Sex, Sab e
Dom"};
int i, tamanho;
pstr = dia;
tamanho = strlen(dia);
clrscr( );
for(i=0;i< tamanho;i++)
 { putch(*( pstr + i) );
  }
getch( );
```

d)
```
char *pstr_e, dia_e[ ] = {"os dias da
semana são: Seg, Ter, quar, Qui, Sex,
sab e Dom"};
int i_e;
pstr_e = dia_e;
clrscr( );
while(*pstr_e)
 { putch (*pstr_e);
   pstr_e += 1;
}
getch( );
```

e)
```
char *dias[ ]={"Segunda", "Terca",
"Quarta", "Quinta", "Sexta",
"Sabado", "Domingo"};
 int i;
 clrscr( );
 for(i = 0; i < 7; i++)
 { printf("\n dia da semana: %d \n", i +
1);
   while(*dias[i])
   { printf("%c", *dias[i]++);
   }
 }
getch( );
```

f)
```
char *dias[ ]={"Segunda", "Terca",
"Quarta", "Quinta", "Sexta",
"Sabado", "Domingo"};
 int i;
 clrscr( );
for(i=0;i<7;i++)
     { printf("\n dia da semana: %d
\n",i+1);
     printf("\n %s",dias[i]);
    }
```

g)
```
    int *p, i = 20;
    p = &i;
    printf("%i" , sizeof(p));
    printf("%i" , sizeof(*p));
```

h)
```
.............
 struct teste{ int numero;
                float real;
                  char nome[10];
                  char rua[30];
       };

void main(void)
{ struct teste x = {1, 2.5, "Smith",
"Paris" },*p;

   int cont;
   p = &x;
   printf("\n no endereço: %05X está
uma estrutura cujo tamanho é: %i",
p,sizeof(*p));
.....
}
```

i)
```
.....................
struct teste1{ char nome[10];
          char rua[30];
     };
struct teste{ int numero;
              float real;
               s truct teste1 y;
      };
void main(void)
{ struct teste x = {1, 2.5, {"Smith",
"Paris"} },*p;
  int cont;
  p = &x;
      printf("\n no endereco: %05X
tamanho da estrutura: %i", p,
sizeof(*p));
  printf("\n numero: %i", p->numero);
  printf("\n real: %f", p->real);
  printf("\n nome: %s", p->y.nome);
  printf("\n nome: %s", p->y.rua);
  fflush(stdin);
  getche();
}
```

j)

```
struct rotulo_data_parc{
                        char mês[2];
             char separador1;
             char dia[2];
             char separador2;
             char ano[5];
            };

union rotulo_data{          char data_completa[11];
                          struct rotulo_data_parc
data_parcial;
      };
void main(void)
{union rotulo_data data={“20/03/2001”}, *p;
   int cont;
   p = &data;
   printf(“\n exibe a data pelo acesso a string
\”data_completa\”: %s”, \
            data.data_completa);
   data.data_parcial.separador1 = ‘-‘;
   data.data_parcial.separador2 = ‘-‘;

   printf(“\n exibe nova data pelo acesso a string
\”data_completa\”: %s”, \
            data.data_completa);

   p->data_parcial.separador1 = ‘@’;
   p->data_parcial.separador2 = ‘@’;

   printf(“\n exibe novo formato: %s”,
            data.data_completa);

}
```

k)

```
union teste{int numero;
                float real;
                  char nome[10];
                  char rua[30];
       };

void main(void)
{union teste data, *p;
   p = &data;
   printf(“\n  o tamanho da união é: %i”, sizeof(*p));
.....
}
```

l)

```
enum mês{ini,JAN, FEV, MAR, ABR, MAI, JUN, JUL,
AGO, SET, OUT, NOV, DEZ};
void main ()
{
enum mês index;
int resp;
char *meses[13] = {“ini”,”Janeiro”, “Fevereiro”, “Marco”,
“Abril”, “Maio”, “Junho”, “Julho”, “Agosto”,  “setembro”,
“Outubro”, “Novembro”, “Dezembro”};
for (index = JAN; index < =DEZ; index++)
    printf(“\n%s”, meses[index]);
}
```

m)

```
main()
{ float a[ ] = {10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 10.7,
10.8, 10.9,11.0}, cont;
  long int tamanho;
   void *ppp; /* ponteiro para void */
   ppp = a;
   tamanho = sizeof(a)/sizeof(float);
   for(cont = 0; cont < tamanho; cont++)
   {
     /* printf(“%i”, (int )ppp ++); ← não aceita  */

     printf(“%f”, *(int *)ppp );
     (int *)ppp += 1;

   }
   getch();
}
```

n)

```
main()
{ char dias[ ]={“segundatercaquarta”};
   void *ppp; /* ponteiro para void */

   ppp = dias;

   while(*(char *)ppp != ‘\0’)
   { putch(*(char *)ppp);
     (char *)ppp +=1;
   }

   getch();

}
```