EXERCÍCIOS

Para os próximos exercícios, considere que int e float ocupam 4 bytes; Double ocupa 8 bytes e char ocupa 1 byte. Ainda, suponha que o programa será compilado e executado em um sistema de 32 bits, portanto as variáveis ponteiros sempre ocupam 4 bytes (em um sistema 64 bits, os ponteiros ocupam 8 bytes). Considere, também, que a alocação de memória é feita sequencialmente, em ordem decrescente.

Preencha a tabela abaixo, conforme o programa for executado e escreva quais saídas serão apresentadas na tela. Considere que a primeira variável será endereçada em 0055FF10.

```
float a,b;
float *c,*d,*e;
a=40;
b=20;
c=&a;
d=&b;
printf("\n%p%f",&a,a);
printf("\n%p%f",&b,b);
printf("\n%p %p %f",&c,c,*c);
printf("\n%p%p %f",&d,d,*d);
a = *c + *d + 1;
e=c;
c=d;
d=e;
*c = *d + *c + 2;
printf("\n%p%f",&a,a);
printf("\n%p%f",&b,b);
printf("\n%p %p %f",&c,c,*c);
printf("\n%p %p %f",&d,d,*d);
```

Identificador	Endereço	Valor
a	0055FF10	61.0
b	0055FF0C	83.0
c	0055FF08	0055FF0C
d	0055FF04	0055FF10
e	0055FF00	0055FF10

Saídas apresentadas na tela: (Obs: não tem espaço em alguns printf do código)

```
0055FF1040.000000
0055FF0C20.000000
0055FF08 0055FF10 40.000000
0055FF04 0055FF0C 20.000000
0055ff1061.000000
0055FF0C83.000000
0055FF08 0055FF0C 83.000000
0055FF04 0055FF10 61.000000
```

Preencha a tabela da próxima página, conforme o programa for executado e escreva quais saídas serão apresentadas na tela. Considere que a primeira variável será endereçada em 005F0010.

```
float a,b;
float *c,*d,*e;
double f,*g;
a=10;
b=20;
c=&a;
d=&b;
g=&f;
*g=10;
printf("\n%p %f",&a,a);
printf("\n%p %f",&b,b);
printf("\n%p %p %f",&c,c,*c);
printf("\n%p %p %f",&d,d,*d);
printf("\n%p %p %f",&g,g,*g);
e=c;
c=d;
d=e;
*g = *c + *d + *e + a + f+1;
printf("\n%p %f",&a,a);
printf("\n%p %f",&b,b);
printf("\n%p %p %f",&c,c,*c);
printf("\n%p %p %f",&d,d,*d);
printf("\n%p %p %f",&g,g,*g + 1);
```

Identificador	Endereço	Valor
a	005F0010	10.0
b	005F000C	20.0
c	005F0008	005F000C
d	005F0004	005F0010
e	005F0000	005F0010
f	005EFFFC	61.0
g	005EFFF4	005EFFFC

Saídas na tela:

```
005F0010 10.000000
005F000C 20.000000
005F0008 005F0010 10.000000
005F0004 005F000C 20.000000
005EFFF4 005EFFFC 10.000000
005F0010 10.000000
005F000C 20.000000
005F0008 005F000C 20.000000
005F0004 005F0010 10.000000
005EFFF4 005EFFFC 62.000000
```

Preencha a tabela da próxima página, conforme o programa for executado e escreva quais saídas serão apresentadas na tela. Considere que a primeira variável será endereçada em 00552100.

```
float x,d;
float *v,*w,*t;
x=10;
d=20;
w=&x;
t=&d;
printf("\n%p%f",&x,x);
printf("\n%p%f",&d,d);
printf("\n%p %p %f",&w,w,*w);
printf("\n%p %p %f",&t,t,*t);
d = *t + x;
v=w;
```

```
w=t;
t=v;
*t = *v + 5;
printf("\n%p%f",&x,x);
printf("\n%p%f",&d,d);
printf("\n%p %p %f",&w,w,*w);
printf("\n%p %p %f",&t,t,*t);
```

Identificador	Endereço	Valor
x	00552100	15.0
d	005521FC	30.0
v	005521F8	00552100
w	005521F4	005521FC
t	005521F0	00552100

Saída na tela: (Obs: alguns nao tem espaço no printf)

```
0055210010.000000
005521FC20.000000
005521F4 00552100 10.000000
005521F0 005521FC 20.000000
0055210015.000000
005521FC30.000000
005521F4 005521FC 30.000000
005521F0 00552100 15.000000
```

Considere o trecho de código abaixo:

```
floata,b;
float *c, *d,*e;
double f;
double *g;
a=1;b=2;f=3;
```

Identifique quais atribuições abaixo não são recomendadas (ou mesmo, que não podem ser realizadas).

```
f=10;
a=b;
a=&b;
a=*b;
a=c;
a=&c;
a=*c;
a=g;
&a=c;
c=a;
c=&a;
c=*a;
c=d;
c=&d;
c=*d;
c=g;
*c=a;
*c=&a;
*c=*a;
*c=10;
*c=d;
*c=*d;
*c=&d;
*c=*g;
*g=*c;
g=c;
```

Respostas:

Supondo que ”floata,b;” na verdade está escrito como “float a,b;” para não causar erro na execução do código para seja possível responder essa atividade.

Válidas:

```
f=10; a=b; a=*c; c=&a; c=d; *c=a; *c=10; *c=*d; *c=*g;
```

Inválidas:

```
a=&b; //a nao eh ponteiro
```

```
a=*b; //b nao eh ponteiro
a=c; //a nao pode receber endereço
a=&c; //a nao eh ponteiro
&a=c; //nao se pode atribuir algo a um endereço
c=a; // ponteiro nao pode receber tipo float
c=*a; //a nao eh ponteiro
c=*d; //c nao pode receber o valor (float) que b aponta
*c=&a; //c aponta para float
*c=*a; //a nao eh ponteiro
*c=d; //a variavel que c aponta nao pode receber endereços
*c=&d; //a variavel que c aponta n pode receber endereços de ponteiros
a=g; //a nao recebe endereços
```

Não recomendadas:

```
c=&d; //c nao eh um ponteiro de ponteiro
c=g; //c e g sao ponteiros para tipos diferentes
*g=*c; //perde a precisao
g=c; // c e g sao ponteiros para tipos diferentes
```

Obs: caso não saiba a resposta, tente usar a tabela para encontrar a explicação. Lembre-se de perguntar: o valor a ser atribuído é do mesmo tipo? Se a atribuição for feita a uma variável ponteiro, então você deve fazer duas perguntas: o valor atribuído é um endereço de memória? Este endereço de memória permite armazenar valores do tipo declarado no ponteiro?

Obs2: caso não consiga responder, teste no DevC ou CodeBlock para encontrar a resposta. Uma vez que obtenha a resposta, tente achar a explicação. Mas não se esqueça de anotar qual atribuição você teve dificuldade (e precisou da ajuda do DevC ou Codeblock) e envie email para pedir a explicação e confirmar sua resposta.