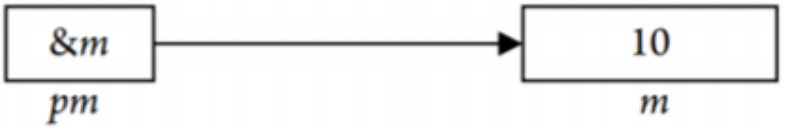
PROVA OPTATIVA - LINGUAGEM DE PROGRAMAÇÃO 2

Aluno: Daniel Sant' Anna Andrade
Matrícula: 20200036904

1)

```
int m,n;
int *pm;

m=10;
pm=&m;
n=*pm;
```



linha 1: declara as variáveis do tipo inteiro m e n;
linha 2: declarava a variável ponteiro do tipo inteiro;
linha 3: inicializa a variável m com o valor de 10;
linha 4: inicializa a variável pm com o endereço de m;
linha 5: inicializa a variável n com o valor armazenado na variável ponteiro *pm, que possui o endereço de m, ou seja a variável n irá inicializar com o valor de 10.

O diagrama demonstra que a variável pm, possui o endereço da variável m, que foi inicializada com o valor de 10.

2)

a) É um for dentro de um for.
Na linha 7 tem o primeiro for está sendo inicializado com "i" recebendo o valor de 0, e esse "i" irá ser incrementado de um em um sempre que todo o seu conteúdo for executado por completo, enquanto "i" for menor que M.
Na linha 8 tem o segundo for é semelhante ao primeiro, ele está sendo inicializado com "j" recebendo o valor de 0, e esse "j" irá ser incrementado de um em um sempre que todo o seu conteúdo for executado por completo, enquanto "j" for menor que N.
Na linha 9 temos a variável "a" que é uma array bidimensional, onde "i" é a linha e "j" a coluna, que estão indicando o índice na array, e a soma de "i" e "j" está sendo alocado nesse índice. Exemplo: No momento em que o "i" estiver valendo 1 e "j" estiver valendo 2, a array "a" irá receber na posição a[1][2] o valor de 3;

```
7         for(i=0;i<M;i++)
8             for(j=0;j<N;j++)
9                 a[i][j]=i+j;
```

b) Semelhante a primeira, porém ao invés de alocar valores nas posições da array “a”, na linha 14 está dando um printf, demonstrando a posição e o valor de cada posição, imprimindo na tela um valor ao lado do outro (obs.: tem um espaço no final de cada printf para que não fique tudo colado na hora da impressão no console). Assim que “j” tiver um valor igual a N, será executado a linha 15, que é para pular linha, pois a próxima linha de valores da array “a” será impressa agora, e isso se repete até que “i” seja igual a M; A linha 16 indica o final do bloco de código dentro do primeiro for.

```
12      for(i=0;i<M;i++) {
13          for(j=0;j<N;j++)
14              printf("a[%d][%d]=%d ",i,j,a[i][j]);
15          printf("\n");
16      }
```

c) Novamente semelhante as linhas 7 e 8 temos um for dentro de um for com o mesmo funcionamento das outras duas. O diferente aqui é na linha 20 onde ele irá adicionar a variável res (que foi inicializada antes na linha 5 com o valor de 0), os valores de cada posição da array “a”. += aqui significa, res = res + a[i][j], ou seja, ele recebe o valor de res, soma com o valor de a[i][j] e aloca esse valor na variável res. Isso irá fazer com que todos os valores alocados na array “a” sejam somados. Na linha 21 é demonstrado o valor final da variável res.

```
18      for(i=0;i<M;i++)
19          for(j=0;j<N;j++)
20              res+=a[i][j];
21      printf("Resultado: %d\n",res);
```

3)

a) A linha 20 tem o primeiro for está sendo inicializado com “t” recebendo o valor de 0, e esse “t” irá ser incrementado de um em um sempre que todo o seu conteúdo for executado por completo, enquanto “t” for menor ou igual a 10.
Na linha 21, está passando o valor do retorno da função “f”, que recebe o parâmetro “t”, para a variável do tipo inteiro funval.
Na linha 22, está pegando a array “plot” e está passando como índice o valor de funval, e nessa posição da array “plot” está sendo colocada o valor “*”.
Na linha 23, está está indicando o fim da array “plot” (\0) na posição funval+1, terminando a array uma posição depois de onde foi alocado a “*”.
Na linha 24, está printando na tela com o printf, o valor de “t” e array “plot”, demonstrando onde está alocado a “*”.
Na linha 25, está limpando a “*” que tinha sido alocada na posição “funval” da array “plot”.
Na linha 26, está limpando o “\0” que indicaria o final da array “plot”.
A linha 27 indica o final do bloco de código que é executado no for.

```
20          for  (t = 0;  t <= 10;  ++t) {
21              funval = f(t);
22              plot[funval] = '*';
23              plot[funval + 1] = '\0';
24              printf("t=%2d%s\n", t, plot);
25              plot[funval] = ' ';
26              plot[funval + 1] = ' ';
27          }
```

b) Esse conjunto de linhas é uma função. Na linha 31 inicia a função “f” do tipo inteiro, que recebe o parâmetro t do tipo inteiro. Na linha 32 se tem o retorno da função “f” que é uma expressão do segundo grau, onde será utilizado o parâmetro “t”. A linha 33 indica o fim do conteúdo da função “f”.

```
31   int f (int t) {
32          return (t * t - 4 * t + 5);
33   }
```