1. Compile e execute o seguinte programa. Comente o que faz cada linha do código:

```
int main (void) {
   typedef struct {
      int dia, mes, ano;
   } data;
   printf ("sizeof (data) = %d\n",
            sizeof (data));
   return EXIT_SUCCESS;
}
```

2. Compile e execute o seguinte programa. Comente o que faz cada linha do código. (O cast (long int) é necessário para que &i possa ser impresso no formato %ld. É mais comum imprimir endereços em notação hexadecimal, usando formato %p, que exige o cast (void *).)

```
int main (void) {
   int i = 1234;
   printf (" i = %d\n", i);
   printf ("&i = %ld\n", (long int) &i);
   printf ("&i = %p\n", (void *) &i);
   return EXIT_SUCCESS;
}
```

3. Compile e execute o seguinte programa. Comente o que faz cada linha do código.

```
int main (void) {
   int i; int *p;
   i = 1234; p = &i;
   printf ("*p = %d\n", *p);
   printf (" p = %ld\n", (long int) p);
   printf (" p = %p\n", (void *) p);
   printf ("&p = %p\n", (void *) &p);
   return EXIT_SUCCESS;
}
```

4. Compile e execute o seguinte programa. Comente o que faz cada linha do código.

```
#include <stdio.h>

int main ()
{
  int *p ;
  int a = 231 ;
  int b = 7680 ;

  printf ("&a vale %p\n", &a) ;
  printf ("&b vale %p\n", &b) ;
  printf ("&p vale %p\n", &p) ;

  printf ("p vale %p\n", p) ;
```

```
  p = &a ;
  printf ("p vale %p\n", p) ;
  printf ("*p vale %d\n", *p) ;

  p = &b ;
  printf ("p vale %p\n", p) ;
  printf ("*p vale %d\n", *p) ;

  *p = 500 ;
  printf ("b vale %d\n", b) ;

  return 0 ;
}
```

5. Compile e execute o seguinte programa. Comente o que faz cada linha do código.

```
#include <stdio.h>
#include <conio.h>
int main(void)
{
  int valor = 27;

  int *ptr;

  ptr = &valor;

  printf("Utilizando ponteiros\n\n");
  printf ("Conteudo da variavel valor: %d\n", valor);
  printf ("Endereço da variavel valor: %x \n", &valor);
  printf ("Conteudo da variavel ponteiro ptr: %x", ptr);

  getch();
  return(0);
}
```

6. Compile e execute o seguinte programa. Comente o que faz cada linha do código.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int x, *ptrx, **pptrx;

  x = 0;
  printf("\nvalor de x = %d\n", x);
  printf("Endereco de x: %x\n\n",&x);

  ptrx = &x;
  pptrx = &ptrx;
```

```
  *ptrx = *ptrx + 10;
  printf("\nvalor de x = %d", x);
  printf("\nEndereco apontado por ptrx: %x\n",ptrx);
  printf("Valor da variavel X que está sendo apontada por ptrx:
%d\n",*ptrx);
  printf("Endereco de memoria da variável ptrx %x\n",&ptrx);

  **pptrx = **pptrx + 10;
  printf("\n\nvalor de x = %d", x);
  printf("\nEndereco apontado por **pptrx: %x",pptrx);
  printf("\nValor da variavel para a qual pptrx faz referencia:
%d",**pptrx);
  printf("\nEndereco de memoria da variavel **pptrx
%x\n\n",&pptrx);

  return 0;
}
```

7. Além de acessar os dados de uma área alocada dinâmicamente como se fosse um vetor, é possível acessá-los através do próprio ponteiro, utilizando a técnica chamada aritmética de ponteiros. Compile e execute o seguinte programa. Comente o que faz cada linha do código.

```
#include <stdlib.h>

void main() {
   int *vet;
   int *ptr;
   vet = (int*)malloc(sizeof(int)*10);
   veterinário[4] = 44;
   ptr = vet;
   *ptr = 11;
   ptr++;
   *ptr = 12;
  printf("%p\n",ptr);
  printf("%d\n", *ptr);
}
```

8. Escreva um programa que leia 10 inteiros da entrada padrão, armazene-os em um vetor e os escreva na saída padrão na ordem contrária a de leitura; todos os acessos ao vetor devem ser feitos usando somente ponteiros, sem usar índices de vetor (vet[i], etc).

9. Escreva um programa para concatenar duas strings usando somente ponteiros.

10. Escreva um programa para calcular o tamanho de uma string usando somente ponteiros.