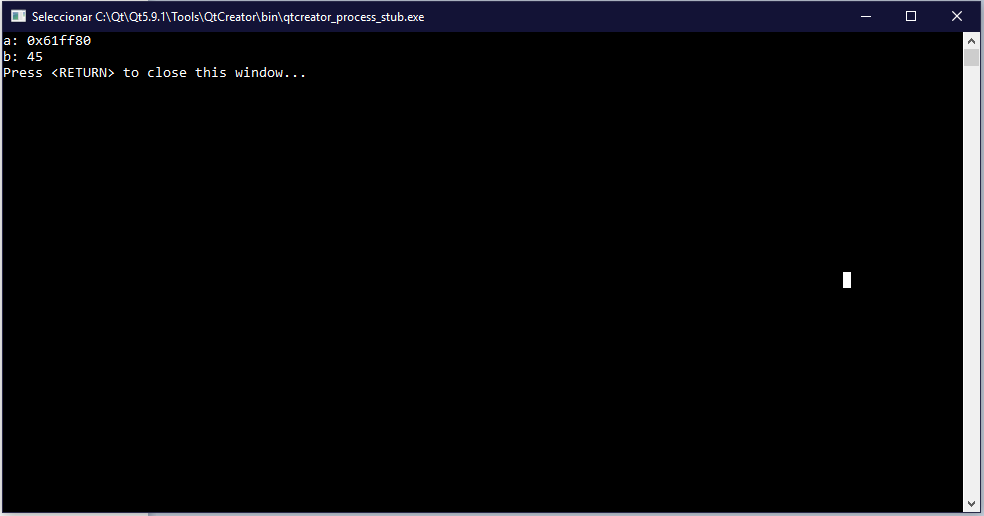
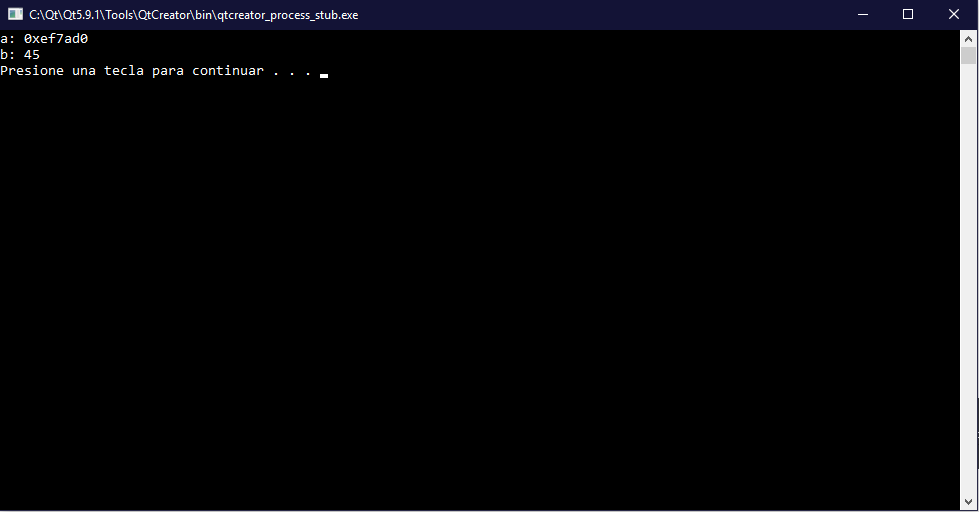
    //1¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a;  
    int b = 45;  
    cout <<"a: " <<a << endl <<"b: " <<b <<endl;  
      
  


Explicación:

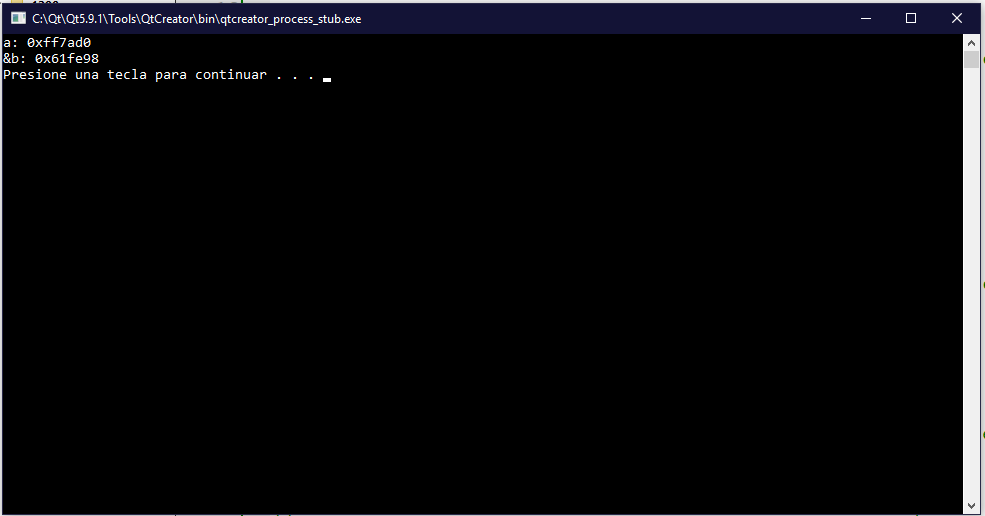
Imprime la dirección de memoria que guarda a e imprime el contenido de b.

    //2. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 45;  
    cout <<"a: " <<a << endl <<"b: " <<b <<endl;  
     
  
  
Explicación:

Apuntador de tipo int declarado como a inicializado con 45. B declarado como entero inicializado como 45. Imprime a como su dirección de memoria que guarda y se imprime b como su valor.

Imprime la posición que guarda a y b imprime el valor guardado

    //3. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 45;  
    cout <<"a: " <<a << endl <<"&b: " <<&b <<endl;

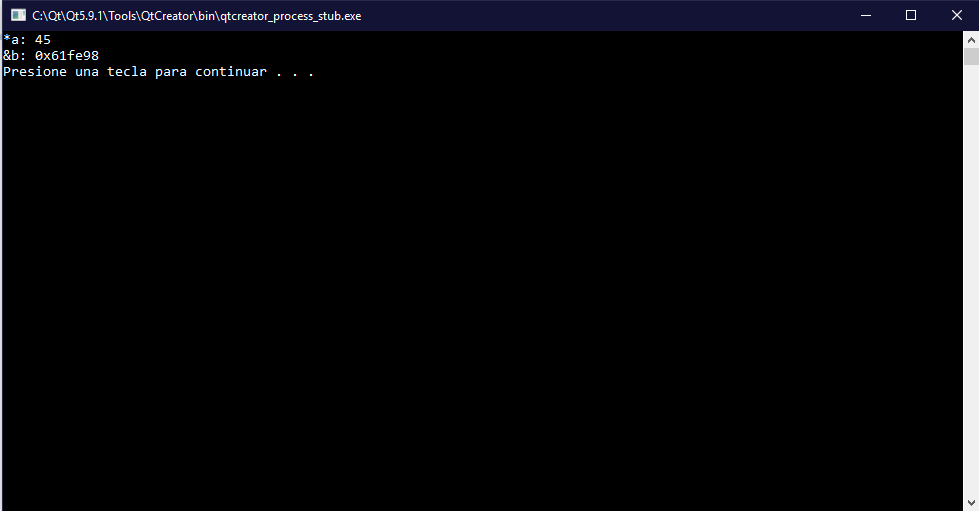


Explicación:

Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 45.

Imprime la dirección que guarda a y imprime la dirección de memoria de la variable b.

    //4. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 45;  
    cout <<"\*a: " <<\*a << endl <<"&b: " <<&b <<endl;

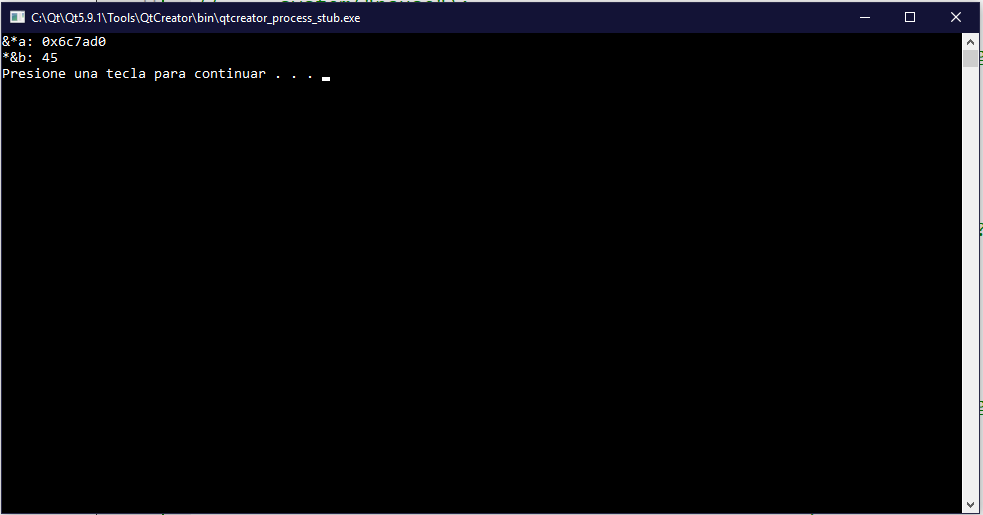


Explicación:

Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 45.

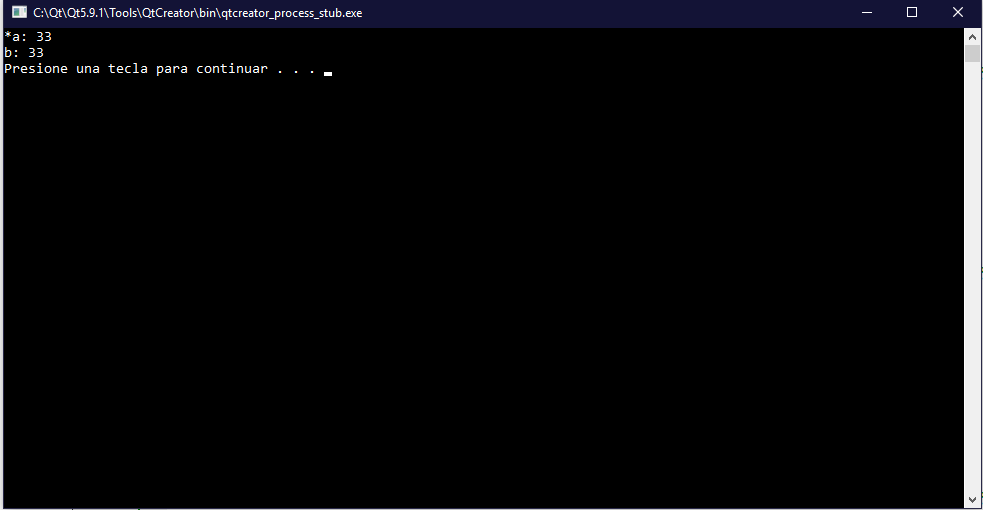
Imprime lo que guarda a y la dirección de memoria de la variable b.

        
    //5. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 45;  
    cout <<"&\*a: " <<&\*a << endl <<"\*&b: " <<\*&b <<endl;

  
  
    Explicación:

Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 45.

Imprime a b como lo que apunta la dirección de b. y a es la posición de memoria de lo que apunta a.

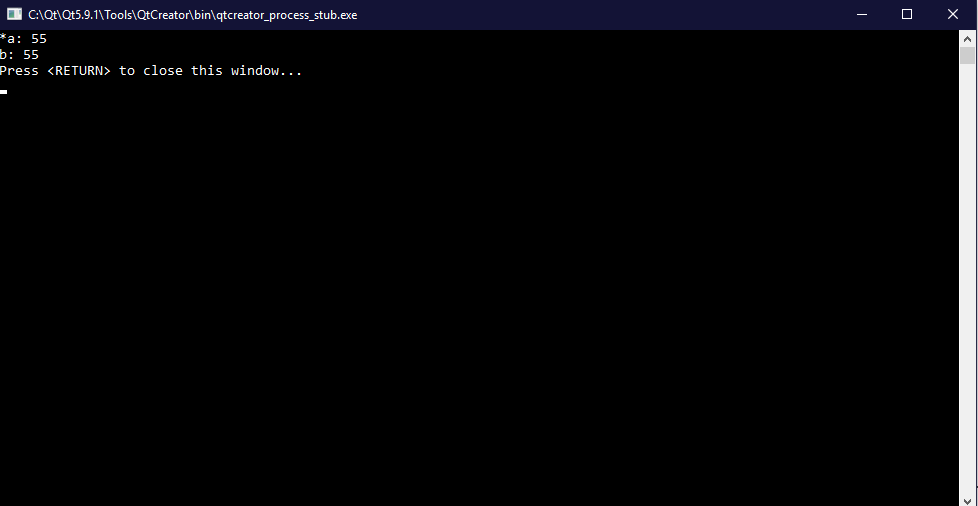
    //6. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 33;  
    a = &b;  
    cout <<"\*a: " <<\*a << endl <<"b: " <<b <<endl;  
  
  


Explicación:

Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 33.

Se guarda en a la posición de memoria de lo que guarda b.

Imprime lo que guarda a como valor y imprime lo que guarda b.

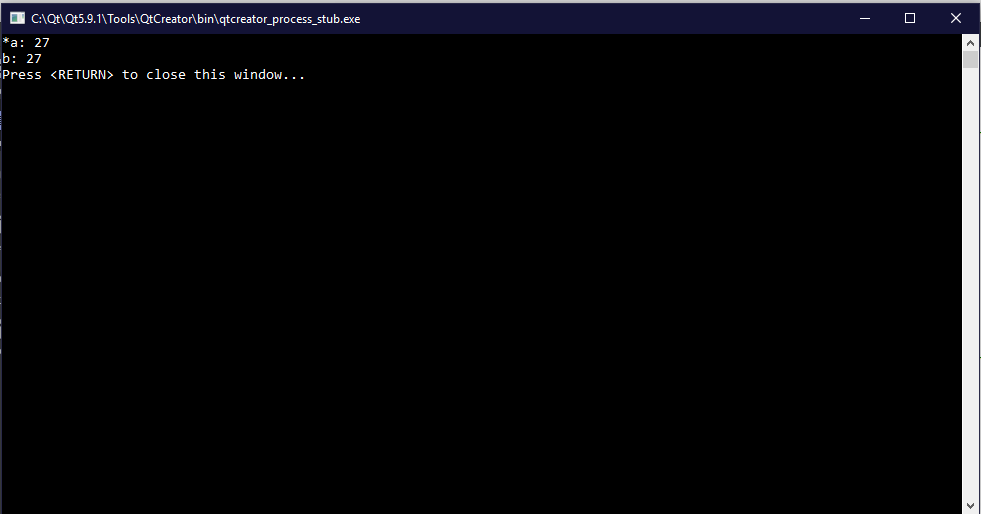
    //7. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 33;  
    a = &b;  
    b = 55;  
    cout <<"\*a: " <<\*a << endl <<"b: " <<b <<endl;  
  
  
Explicación:

Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 33.

Guarda a la dirección de memoria de b.

B cambia su valor

A imprime a lo que apunta y b su valor.

    //8. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 33;  
    a = &b;  
    b = 55;  
    \*a = 27;  
    cout <<"\*a: " <<\*a << endl <<"b: " <<b <<endl;  
  


Explicación:

Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 33.

Guarda a la dirección de memoria de b.

B cambia su valor

A cambia el valor de lo que apunta modificando directamente a b.

Imprime a lo que apunta y b su valor.

    //9. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 33;  
    a = &b;  
    b = 55;  
  
    a = new int(27);  
    cout <<"\*a: " <<\*a << endl <<"b: " <<b <<endl;  
      


Explicación:

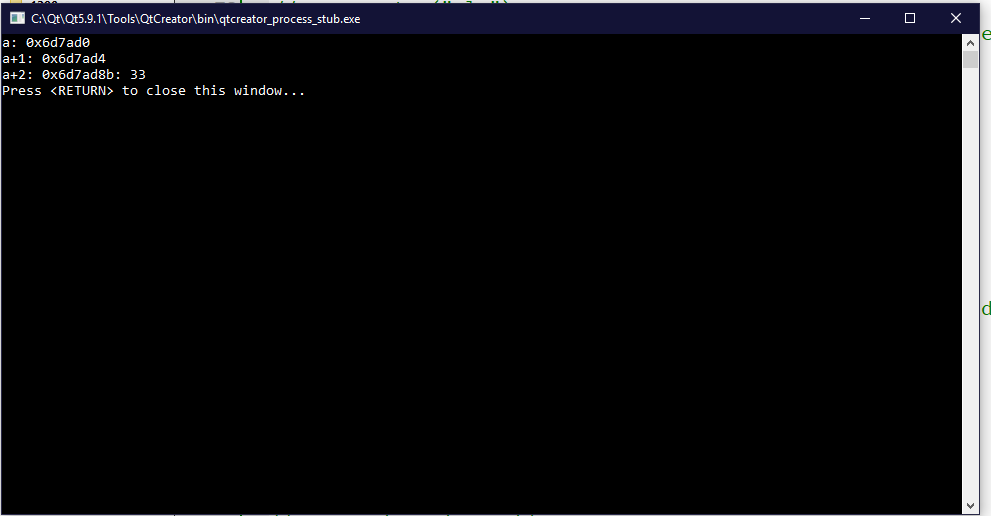
Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 33.

Guarda a la dirección de memoria de b.

B cambia su valor

A asigna un nuevo valor a lo que apunta

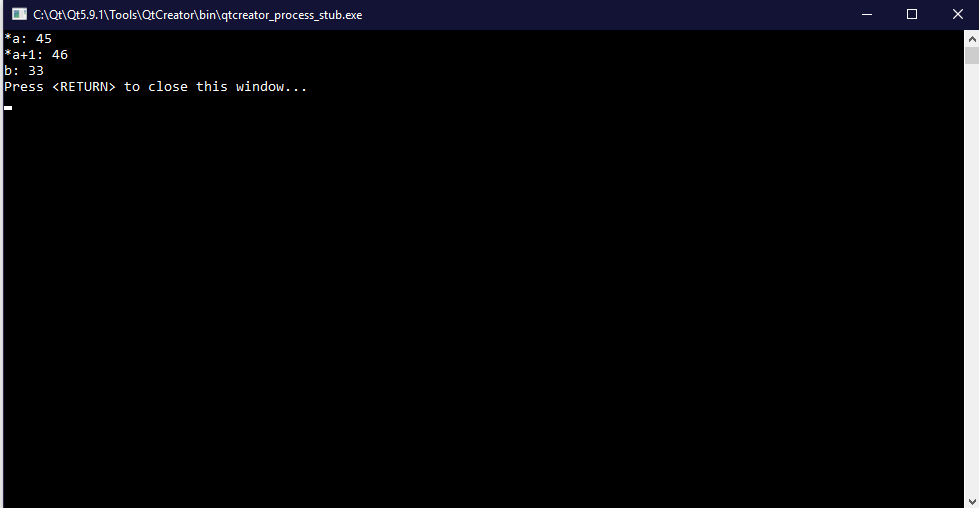
Imprime lo que apunta a y b imprime su valor.

      
    //10. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 33;  
    cout <<"a: " <<a << endl <<"a+1: " <<a+1 <<endl<< "a+2: "<< a+2 <<"b: " <<b <<endl;  
    //10.1 ¿Qué resultará de evaluar a+2?  
      
  
Explicación:

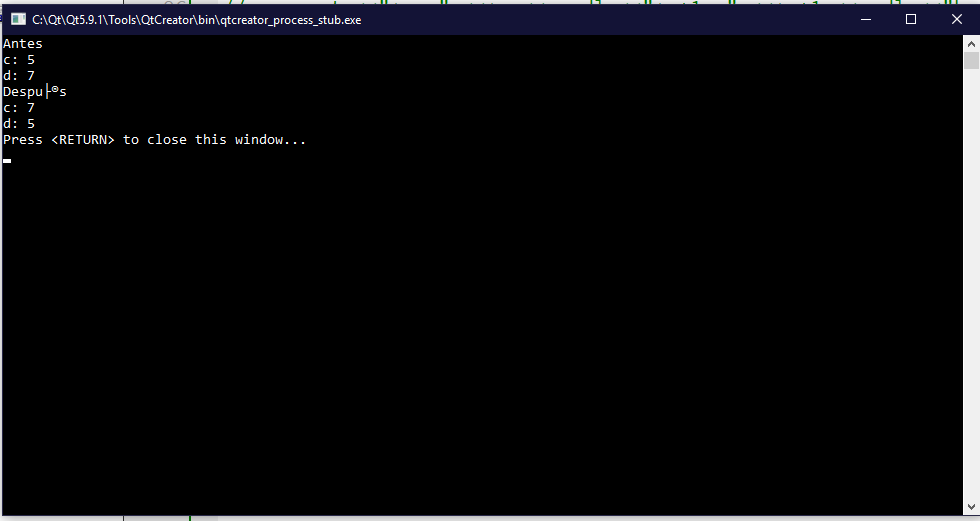
Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 33.

A imprime su dirección que esta guardando. A+1 imprime un salto de bits en la memoria de en este caso de tipo int (4 bytes). A+2 imprime un salto de dos espacios de tipo entero (8 bytes)

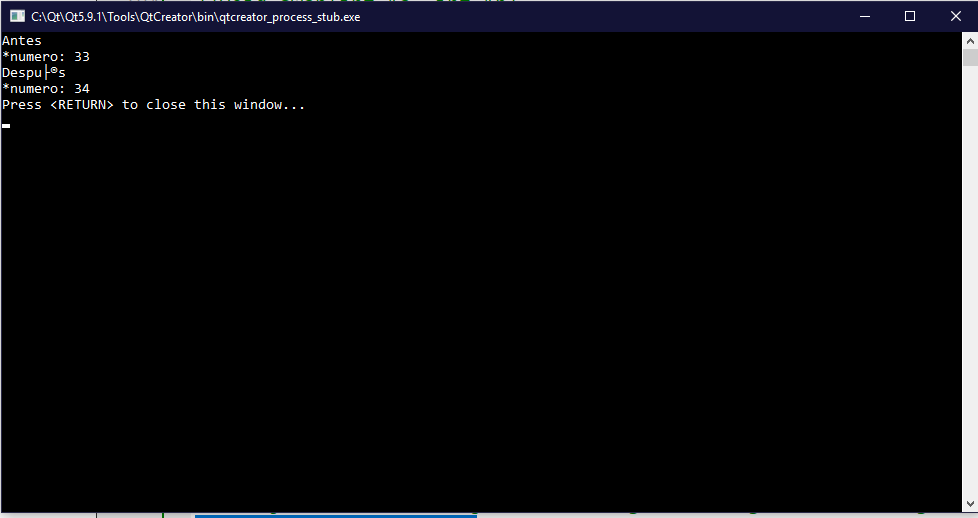
B imprime su valor.

    //11. ¿Qué resulta de ejecutar el siguiente segmento de código?  
    int\* a = new int(45);  
    int b = 33;  
    cout <<"\*a: " <<\*a << endl <<"\*a+1: " <<\*a+1 <<endl <<"b: " <<b <<endl;  
      
  
  
  
Explicación:

Se declarara como apuntador de tipo entero y se inicializa como 45. Se declara b como entero y se inicializa como 33.

//12. ¿Qué resulta de ejecutar el siguiente segmento de código?  
void swap(int &a, int &b)  
{  
    int temporal;  
    temporal = a;  
    a = b;  
    b = temporal;  
}  
  
int main()  
{  
    int c = 5;  
    int d = 7;  
    cout <<"Antes" <<endl <<"c: " <<c <<endl <<"d: " <<d <<endl;  
    swap(c, d);  
    cout <<"Después" <<endl <<"c: " <<c <<endl <<"d: " <<d <<endl;  
    return 0;  
}  
  


Explicación:

  
  
//13. ¿Qué resulta de ejecutar el siguiente segmento de código?  
void incrementa(int \*a)  
{  
    (\*a)++;  
}  
int main()  
{  
    int\* numero = new int(33);  
    cout <<"Antes" <<endl <<"\*numero: " <<\*numero <<endl;  
    incrementa(numero);  
    cout <<"Después" <<endl <<"\*numero: " <<\*numero <<endl;  
    return 0;  
}

Explicación: