Apellido y Nombre: …… *Fecha: ……/……/……*

CONCEPTOS A TENER EN CUENTA

CODIFICACIÓN EN LENGUAJE C/C++

EQUIVALENCIAS ENTRE PSEUDOCÓDIGO Y LENGUAJE C/C++.

ASIGNACIÓN	
suma ← suma + 10	suma=suma + 10;
LECTURA	
leer valor	cin >> valor;
ESCRITURA	
escribir "hola mundo!!!"	cout << "hola mundo!!!";
CONDICIONALES O SELECTIVAS SIMPLES	

si condición entonces
  acciones
fin_si
	if (condición)
	  acciones;

CONDICIONALES O SELECTIVAS DOBLES	

si condición entonces
  acciones1
sino
  acciones2
fin_si
	if (condición)
	  acciones1;
	else
	  acciones2;

CONDICIONALES O SELECTIVAS MÚLTIPLES	

según opción hacer
  op1: acciones_1
  op2: acciones_2
  …
  opn: acciones_n
de otro modo
  acciones
fin_segun
	switch (opción)
	{
	  case op1: acciones_1; break;
	  case op2: acciones_2; break;
	  …
	  case opn: acciones_n; break;
	  default: acciones;
	}

REPETIR	

Repetir
  acciones
hasta_que condición
	do
	{
	  acciones;
	} while (condición);

MIENTRAS	

mientras condición hacer
  acciones
fin_mientras
	while (condición)
	{
	  acciones; }

PARA	

para v desde vi hasta vf con paso n hacer
  acciones
	for (v=vi; v<=vf; v++)
	{
	  acciones; }

```
fin_para
```

PROGRAMAS EN C. ESTRUCTURA GENERAL

Un programa codificado en lenguaje *C* se organiza, básicamente, en 3 secciones

- Declaraciones
  - Librerías del lenguaje: la directiva *#include* permite indicar al compilador qué librerías debe incluir en el programa objeto para generar el programa ejecutable correspondiente. Las librerías a utilizar se indican entre paréntesis angulares, por ejemplo, *<stdio.h>* (librería de las funciones estándar de entrada/salida). De esta manera, el programador puede utilizar las funciones internas del lenguaje.
  - Módulos del programador: se especifica el tipo y argumentos de los módulos escritos por el programador.
  - Variables globales: se especifica el tipo de dato y nombre de las variables globales del programa.
  - Constantes: se especifica el tipo, nombre y valor de las constantes del programa.
- Programa principal (función *main*): la función *main()* contiene declaraciones de variables e instrucciones necesarias para controlar la secuencia de operaciones que ejecuta el programa a fin de resolver el problema para el que fue pensado.
- Módulos del programador: se especifica el código correspondiente a cada uno de los módulos creados por el programador. Un módulo contiene la declaración de variables e instrucciones que resuelven un subproblema específico.

A continuación, se presenta un modelo de la estructura general para un programa codificado en lenguaje *C*.

```
/* Comentario inicial: nombre del programa del programador, fecha, etc */

/* Archivos de cabecera (prototipos de funciones de librería) */
#include <archivo_cabecera.h>
#include <archivo_cabecera.h>

// Prototipos de funciones y procedimientos escritos por el programador
tipo_dato modulo1 (argumentos);
tipo_dato modulo2 (argumentos);

/* Variables y constantes globales */
tipo_dato variable_global;

const tipo_dato constante_global=valor;
#define PI 3.14

/* Algoritmo principal */
tipo_dato main(argumentos)
{
   /* Variables locales del algoritmo principal */
   tipo_dato nombre_variable1, nombre_variable2;
   tipo_dato nombre_variable3, nombre_variable4;
   ...
   /* Instrucciones del algoritmo principal */
   ...
   modulo1(argumentos);
   ...
   modulo2(argumentos);
   ...
   return valor;
}

/* Código completo de las funciones escritas por el programador */
tipo_dato modulo1 (argumentos)
{
   /* Variables locales e instrucciones del módulo */
}

tipo_dato modulo2 (argumentos)
{
```

```
        /* Variables locales e instrucciones del módulo */
    }
```

TIPOS DE DATOS BÁSICOS

En un programa en *C*, los datos que se utilicen pueden definirse según los tipos presentados en la siguiente tabla.

Nombre	Descripción	Tamaño	Rango
char	Caracter (código ASCII)	8 bits	Con signo: -128 … 127 Sin signo: 0 … 255
short int (short)	Número entero corto	16 bits	Con signo: -32768 … 32767 Sin signo: 0 … 65535
int	Número entero	32 bits	Con signo: -2147483648 … 2147483647 Sin signo: 0 … 4294967295
long int (long)	Número entero largo	64 bits	Con signo: -9223372036854775808, 9223372036854775807 Sin signo: 0 … 18446744073709551615
float	Número real	32 bits	3,4*10-38 … 3,4*10+38 (6 decimales)
double	Número real en doble precisión	64 bits	1,7*10-308 … 1,7*10+308 (15 decimales)
long double	Número real largo de doble precisión	80 bits	3,4*10-4932 … 1,1*10+4932
bool	Valor booleano	1 bit	true (VERDADERO) o false (FALSO)

La selección del tipo de dato adecuado para los elementos del programa se realiza teniendo en cuenta el rango de valores a representar y las operaciones que se deben aplicar.

FUNCIONES BÁSICAS DE ENTRADA/SALIDA DE DATOS.

Las funciones de entrada/salida estándar de *C* están definidas en la biblioteca *stdio*. Cuando estas funciones se utilizan en un programa fuente es preciso incluir el archivo *stdio.h* mediante la directiva de precompilación *#include <stdio.h>.* Entre las funciones que contiene esta librería se encuentran *printf*, *scanf* y *gets*.

La función *printf* es la salida genérica por consola utilizada por *C*, mientras que la función *scanf* es la entrada estándar asociada al teclado. Tanto la función *printf* como la función *scanf* permiten especificar el formato en el que se van a escribir o leer los datos, esto se conoce como entrada/salida formateada.

La función de entrada *gets* permite almacenar una cadena de caracteres ingresada por teclado. Es decir, lee datos de la entrada estándar y los almacena en la variable de tipo cadena que utiliza como argumento. La sintaxis de esta función es la siguiente:

```
gets(nombre_variable);
```

En *C++* además de las funciones *printf* y *scanf*, que siguen estando vigentes, se pueden utilizar las "funciones" *cin* y *cout*. Para utilizarlas es necesario incluir la librería *iostream* especificando la directiva *#include <iostream>, cin* y *cout* responden a la siguiente sintaxis:

```
cin >> nombre_variable;
cout << "cadena de caracteres" << nombre_variable << endl;
```

Utilizando *cin* y *cout* no es necesario especificar el tipo de dato que se imprimirá o leerá, asociándolo con un formato determinado (como ocurre con *printf* y *scanf*), sino que es el propio programa el que decide el tipo de dato en tiempo de ejecución. De este modo, *cin* y *cout* admiten tanto los tipos predefinidos como aquellos tipos de datos definidos por el usuario. Al ser C++ una ampliación del lenguaje C, es necesario agregar nuevas palabras reservadas. Estas palabras reservadas están en un espacio de nombres o *"namespace"*. Para usar las palabras reservadas *cout* y *cin*, que están el *namespace std* (standard), se debe incorporar la instrucción:

```
using namespace std;
```

al incorporar en la cabecera del programa esta directiva estamos indicando al compilador que use el espacio de nombres *std*, y que

busque e interprete todos los elementos definidos en el archivo.

DOCUMENTACIÓN INTERNA. COMENTARIOS

Los comentarios en un programa fuente *C* pueden especificarse para líneas individuales o párrafos completos. Un comentario de línea se indica con el símbolo // y un comentario de párrafo se especifica con los símbolos /* (inicio del comentario) */ (fin del comentario).

OPERADORES EN C

La siguiente tabla presenta los operadores básicos utilizados en *C*.

Tipo	Operadores
Aritmético	Potencia: ***pow(x,y)*** (librería *math.h*); *x, y* valores numéricos Producto: ***** Cociente: **/** Módulo o resto: **%** Sumar: **+** Diferencia: **-**
Alfanuméricos	Operaciones con cadenas (librería *string.h*) *strcat(s,t);* concatena *t* al final de *s.* *strcmp(s,t);* compara *s* y *t*, retornando negativo, cero, o positivo para: *s<t*, *s==t*, *s > t.* *strcpy(s,t);* copia *t* en *s.* *strlen(s);* retorna la longitud de *s*. donde *s* y *t* son variables de tipo cadena.
Lógicos	Negación (NO, NOT): **!** Conjunción (Y, AND): **&&** Disyunción (O, OR): **||**
Relacionales	Igual: **==** Distinto: **!=** Mayor: **>** Mayor o igual: **>=** Menor: **<** Menor o igual: **<=**
Asignación	**=**

SOFTWARE DE PROGRAMACIÓN EN C++:

Dev-C++ es un entorno de desarrollo integrado para programar en lenguaje C/C++. Usa MinGW, una versión de GCC, como compilador. Dev-C++ puede además ser usado en combinación con Cygwin y cualquier otro compilador basado en GCC. El Entorno está desarrollado en el lenguaje Delphi de Borland. Fuentes y sitio de descarga https://www.bloodshed.net/index.html.

CodeBlocks es un entorno de desarrollo integrado libre y multiplataforma para el desarrollo de programas en lenguaje C++. Está basado en la plataforma de interfaces gráficas WxWidgets, por lo que puede usarse libremente en diversos sistemas operativos, y está bajo GNU. Es una herramienta para desarrollar programas en C++ que ofrece una interfaz sencilla a los usuarios. Fuentes y sitio de descarga http://www.codeblocks.org/downloads/binaries.

Zinjal es un IDE (entorno de desarrollo integrado) libre y gratuito para programar en C/C++. Pensado originalmente para ser utilizado por estudiantes de programación durante el aprendizaje, presenta una interfaz incial muy sencilla, pero sin dejar de incluir funcionalidades avanzadas que permiten el desarrollo de proyectos tan complejos como el propio Zinjal. Fuentes y sitio de descarga: http://zinjai.sourceforge.net/

Opciones online: *http://cpp.sh/, https://www.jdoodle.com/online-compiler-c++, entre otros.*

App: *http://play.google.com/store/apps/details?id=com.duy.c.cpp.compiler*

1. Diseñe un algoritmo (pseudocódigo) que sume 30 valores ingresados por el usuario. Codifique el algoritmo en C, incluyendo las librerías necesarias.

```
PROGRAMA SUMAR_VALORES
VARIABLES
  suma, valor, i: ENTERO
INICIO
      suma ← 0
      PARA i DESDE 1 HASTA 30 HACER
            ESCRIBIR "ingrese valor:"
            LEER valor
            suma ← suma + valor
      FINPARA
      ESCRIBIR "la suma es:" suma
FIN
```

```
//Sumar_Valores
#include <iostream>
#include <stdlib.h>

using namespace std;

main ()
{ int suma,valor,i;
  suma=0;
  for(i=1;i<=30;i++)
  {
    cout << "Ingrese valor:";
    cin >> valor;
    suma=suma+valor;
  }
  cout << "La suma es:" << suma << endl;
  system("pause");
}
```

El algoritmo permite ingresar 30 valores (cantidad fija) y calcular su suma. El ingreso de los valores y cálculo de las sumas parciales se realiza dentro de una estructura *PARA*. Esta estructura permite especificar las acciones a repetir y el número de iteraciones (repeticiones) a realizar. Obsérvese que la variable *suma*, inicializada en cero, funciona como acumulador de los valores de entrada. En el código fuente se especifican el tipo de las variables de entrada, proceso y salida utilizadas, valores de inicialización y operaciones de entrada/salida de acuerdo a la sintaxis del lenguaje.

2. Diseñe un algoritmo (pseudocódigo) que calcule el promedio de valores ingresados por el usuario. Considere que el ingreso finaliza a pedido del usuario. Codifique el algoritmo en C, incluyendo las librerías necesarias.

```
PROGRAMA PROMEDIO_VALORES
VARIABLES
  dato, contador: ENTERO
  suma, promedio: REALES
  respuesta: CARACTER
INICIO
    suma←0
    contador←0
    REPETIR
        ESCRIBIR "Ingrese dato:"
        LEER dato
        suma←suma + dato
        contador←contador + 1
        ESCRIBIR "Ingresar más datos s/n:"
        LEER respuesta
    HASTA_QUE respuesta='n' O respuesta='N'
    promedio <- suma/contador
    ESCRIBIR "Promedio:" promedio
FIN
```

```
//Promedio_Valores
#include <iostream>
#include <stdlib.h>

using namespace std;

main()
{ int contador,dato;
  float promedio,suma;
  char respuesta;
  suma=0;
  contador=0;
  do
   {
       cout << "Ingrese dato:";
       cin >> dato;
       suma=suma+dato;
       contador++;
       cout << "Ingresar mas datos S/N:";
       cin >> respuesta;
   } while ((respuesta!='n') && (respuesta!='N'));
  promedio=suma/contador;
  cout << "El promedio es: " << promedio << endl;
  system("pause");
}
```

El algoritmo permite introducir valores, en tanto el usuario desee continuar con el ingreso (la entrada de datos finaliza a petición del usuario), y calcular su promedio. Al desconocerse a priori la cantidad de números a promediar es necesario utilizar una variable que lleve cuenta del ingreso, en este caso, usando la variable *contador*. Además la variable *suma* funciona como acumulador de los

datos de entrada y la variable *respuesta* permite evaluar si se continua o no con las acciones del bucle *REPETIR*.

En el programa fuente puede observarse que las variables *suma* y *promedio* están definidas como reales (*float*). Es conveniente destacar que el operador de división (/) realiza el cociente entero si los datos que opera son enteros y calcula el cociente real si al menos uno de los operandos es real (**promedio=suma/contador**). Otra observación importante es la condición de finalización de bucle. Mientras que en el pseudocódigo se evalúa la condición **respuesta='n' O respuesta='N'**, en el programa en *C* la condición del bucle es **((respuesta!='n') && (respuesta!='N'))**. Esto se debe a que en DISEÑO la estructura *REPETIR* itera en tanto la condición sea FALSA y finaliza con condición VERDADERA. En tanto que en *C*, la estructura pos-condicional *do-while* repite las acciones del bucle si la condición es VERDADERA y finaliza cuando ésta se hace FALSA.

3. Diseñe un algoritmo (pseudocódigo) que determine el valor máximo de una serie de números enteros ingresados por el usuario. La cantidad de números a considerar es indicada por el usuario. Codifique el algoritmo en C, incluyendo las librerías necesarias.

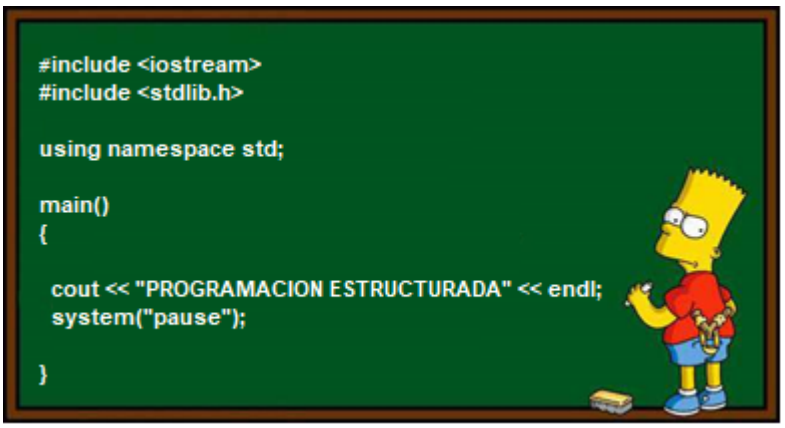
```
PROGRAMA MAXIMO
VARIABLES
  bmax: LOGICO
  cantidad, valor, max, i: ENTERO
INICIO
        ESCRIBIR "Ingrese cantidad de datos: "
        LEER cantidad
        bmax←VERDADERO
        PARA i DESDE 1 HASTA cantidad HACER
          ESCRIBIR "Ingrese valor: "
          LEER valor
          SI bmax=verdadero ENTONCES
                bmax←FALSO
                max←valor
          SINO
              SI valor > max ENTONCES
                  max← valor
              FIN_SI
          FIN_SI
        FIN_PARA
        ESCRIBIR 'el máximo valor es: ', max
FIN
```

```
// Programa Maximo
#include <iostream>
#include <stdlib.h>

using namespace std;

main()
{bool bmax;
 int cantidad,valor,max,i;
 cout << "Ingrese cantidad de valores:";
 cin >> cantidad;
 bmax=true;
 for(i=1;i<=cantidad;i++)
 { cout << "Ingrese valor:";
    cin >> valor;
    if (bmax==true)
     { bmax=false;
       max=valor;   }
    else
     { if (valor>max)
          max=valor;   }
 }
 cout << "El maximo es:" << max << endl;
 system("pause");
}
```

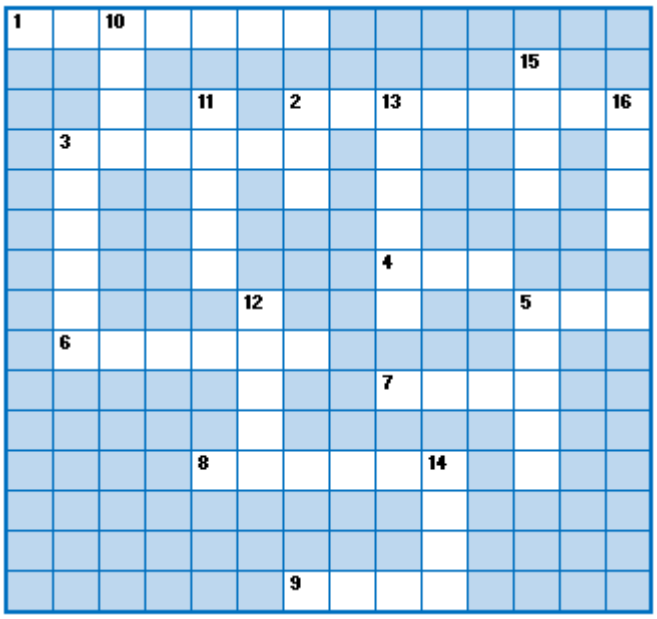
El algoritmo permite determinar el máximo valor de una serie de datos ingresada por el usuario. En este ejemplo puede observarse cómo se usa una variable bandera (*bmax*, variable lógica) para identificar el primer valor de la serie. Este primer valor es asignado a la variable *max*, modificándose entonces *bmax* para omitir la asignación inicial en los siguientes ingresos y comparar el primer máximo con los nuevos datos. Así, sólo los valores introducidos por el usuario serán evaluados para obtener el máximo.

EJERCICIOS A RESOLVER

1. ¿Cuál es el propósito del siguiente programa? Modifíquelo de modo que muestre su nombre, carrera y facultad.



2. Complete el siguiente crucigrama con términos relacionados al lenguaje de programación C/C++.



HORIZONTALES
1) Directiva de compilación que agrega librerías a un programa en C.
2) Librería que proporciona funciones de entrada-salida (C/C++).
3) Función que calcula la longitud de una cadena de caracteres.
4) Función que realiza el cálculo de potencia.
5) Instrucción para la entrada interactiva de datos.
6) Instrucción para la salida de datos (especifica formato de salida).
7) Módulo principal de un programa fuente en C.
8) Tipo de dato real en C (64 bits).
9) Tipo de dato lógico en C.

VERTICALES
2) Tipo de dato entero en C (32 bits).
3) Función que compara cadenas de caracteres.
5) Palabra reservada utilizada para definir constantes.
10) Instrucción para la salida de datos.
11) Tipo de dato real en C (32 bits).
12) Librería asociada a las funciones estándar de entrada-salida (C).
13) Función que permite copiar una cadena de caracteres en otra.
14) Final de línea (para crear un salto siguiente línea).
15) Función de entrada para cadenas de caracteres.
16) Librería asociada a funciones matemáticas.

3. El siguiente programa, codificado en lenguaje C, permite ingresar 2 números enteros y mostrar el mayor de ellos. Codifíquelo y guárdelo como tp5-3.cpp. Compile y ejecute el programa. ¿Cómo modificaría el programa para detectar valores iguales?

```
// librerias que permiten utilizar las funciones del lenguaje
#include <iostream>
#include <stdlib.h>
using namespace std;
// programa principal
main ()
{  int num1,num2;
   cout << "Ingrese primer valor:";
   cin >> num1;
   cout << "Ingrese segundo valor:";
   cin >> num2;
   if (num1 > num2)
      cout << "El mayor es: " << num1 << endl;
   else
      cout << "El mayor es: " << num2 << endl;
   system("pause");
}
```

4. El siguiente programa, codificado en lenguaje C, permite ingresar un carácter y determinar si es una letra mayúscula o minúscula. Codifíquelo y guárdelo como tp5-4.cpp. Compile y ejecute el programa. ¿Cómo modificaría el programa para detectar que se introdujo cualquier otro símbolo?

```
#include <iostream>
#include <stdlib.h>
using namespace std;
main ()
{  char letra;
   cout << "Ingrese caracter:";
   cin >> letra;
   if (letra>='A' && letra letra<='Z')
     cout << letra << "es una mayuscula" << endl;
   else
     if (letra>='a' && letra<='z')
        cout << letra << "es una minuscula" << endl;
   system("pause");
}
```

5. El siguiente programa, codificado en lenguaje C, permite ingresar 2 cadenas de caracteres y determinar si son iguales o no. Codifíquelo y guárdelo como tp5-5.cpp. Compile y ejecute el programa. ¿Cómo modificaría el programa para mostrar la cantidad de caracteres de la cadena de mayor longitud?

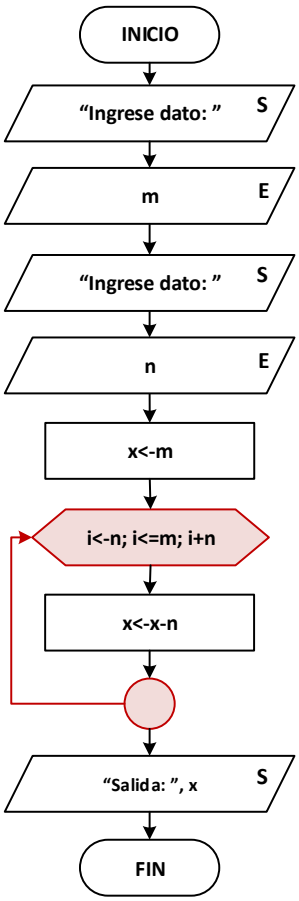
```
#include <iostream>
#include <string.h>
#include <stdlib.h>
typedef char tcad[20]; // definición del tipo cadena
using namespace std;
main()
{  tcad frase1,frase2;
   cout << "Ingrese cadena:";
   gets(frase1);
   cout << "Ingrese cadena:";
   gets(frase2);
   if (strcmp(frase1,frase2)==0)
     cout << "CADENAS IGUALES" << endl;
   else
     cout << "CADENAS DISTINTAS" << endl;
   system("pause");
}
```

6. El siguiente programa, codificado en lenguaje C, permite ingresar un carácter y determinar si es una vocal (mayúscula) o no. Codifíquelo y guárdelo como tp5-6.cpp. Compile y ejecute el programa. ¿Cómo modificaría el programa para que también reconozca minúsculas? Investigue las funciones de conversión de caracteres de C.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
main ()
{  char vocal;
   cout << "Ingrese letra:";
   cin >> vocal;
   switch (vocal)
   {
    case 'A': cout << "VOCAL A" << endl;
              break;
    case 'E': cout << "VOCAL E" << endl;
              break;
    case 'I': cout << "VOCAL I" << endl;
              break;
    case 'O': cout << "VOCAL O" << endl;
              break;
    case 'U': cout << "VOCAL U" << endl;
              break;
    default: cout << "NO ES VOCAL" << endl;
   }
   system("pause");
}
```

7. En el TP4 se propuso realizar el cálculo del producto de 2 valores enteros positivos mediante sumas sucesivas. Entre los esquemas propuestos se encontraba la implementación del bucle de cálculo con estructuras REPETIR aplicando el criterio de finalización por bandera. Traduce este esquema a lenguaje C.
8. Codifique en C un programa que calcule la potencia (mediante productos sucesivos) de un número entero A elevado a otro B, ambos valores ingresados por el usuario. Considerando que los valores pueden ser positivos y/o negativos, controle que el resultado se calcule correctamente. Guarde el archivo fuente como tp5-8.cpp. Compile y ejecute el programa.

9. Un joven que recientemente aprendió a programar recibió la tarea de codificar, en lenguaje C, el siguiente diagrama de flujo. Un amigo más experimentado observó 7 errores en este código fuente. ¿Podrías ayudar a este programador novato a detectar y corregir los errores? ¿Cuál es el propósito del programa?




```
#include <iostream.h>
#include <stdlib.h>
using namespace std;
main()
{   int m,n,x;
    cout << "Ingrese dato:";
    cin >> m;
    cout << "Ingrese dato:";
    cin << n;
    x=m;
    for (i=n,i<=m,i=i+n);
        x=x-n
    cout << "Salida: " x << endl;
    system("pause");
}
```

10. Un compañero te envió un código fuente por WhatsApp, desafortunadamente el mensaje se dañó y parte del código se perdió. Tu precavido compañero también te envió la prueba de escritorio del programa. ¿Podrás restaurarlo? ¿Cuál es su objetivo?

```
#include <iostream>
#include <stdlib.h>
using namespace std;
main()
{ ☹☹☹☹ n,s,☹;
  ☹☹☹☹ << "Ingrese dato: ";
  cin >> n;
  ☹☹☹☹☹;
  for(i=1;i<=☹☹☹;☹☹☹☹)
  { s=s+i;
    ☹☹☹☹☹☹;
    ☹☹☹☹☹☹;
  }
  cout << "Salida: " ☹☹☹☹☹☹;
  system("pause");
}
```

n	s	i	i<=n
4			
	0		
		1	1<=4? VERDADERO
	1		
	2		
	1		
		2	2<=4? VERDADERO
	3		
	5		
	4		
		3	3<=4? VERDADERO
	7		
	10		
	9		
		4	4<=4? VERDADERO
	13		
	17		
	16		
		5	5<=4? FALSO

11. Sabías que el cubo de un número puede calcularse como se muestra en la siguiente pirámide. ¿Podrías diseñar y codificar un algoritmo que calcule el cubo de un número entero positivo aplicando este método?

$$\begin{aligned} 1 &= 1^3 \\ 3+5 &= 2^3 \\ 7+9+11 &= 3^3 \\ 13+15+17+19 &= 4^3 \\ 21+23+25+27+29 &= 5^3 \end{aligned}$$

12. Codifique un programa en C que invierta los dígitos de un número entero. Por ejemplo, si el usuario ingresara el valor 1829, el programa deberá generar como salida el número 9281. Ayuda: en el TP4 descubriste cómo extraer los dígitos de un número.

13. En matemáticas es muy común aplicar el cálculo de factorial para diversas fórmulas. ¿Podrías codificar 3 programas diferentes que realicen este cálculo?

14. En el TP1 descubriste como calcular valores de la siguiente serie de igualdades: **1+4=5 2+5=12 3+6=21 8+11=¿?** ¿Podrías diseñar y codificar un algoritmo que calcule cualquier término de la serie?

15. Codifique en C un programa que calcule la siguiente sumatoria

$$\sqrt{6 \times \sum_{i=1}^{n} \frac{1}{i^2}}$$

Considere que el valor *n* es ingresado por el usuario. Además, sabiendo que el resultado corresponde a un muy conocido número de las matemáticas, ¿cuántos términos deben calcularse para que el resultado tenga una precisión de al menos 4 dígitos fraccionarios?