

SOLUCIONARIO EXAMEN FINAL MB545
CICLO 2021-2

PREGUNTA 1

```
class estadisticas {
protected:
    int* nVector, * nModa, PI, moda;
    float media, mediana, desv_estd;
public:
    void ParImpar(int N);
    void leevector(int N);
    void escribirvector(int N);
    void ordenavector(int N);
    void mostrarvector(int N);
    double Media(int N);
    float Mediana(int N);
    void Moda(int N);
    double DesvEstd(int N);
};

void estadisticas::ParImpar(int N) {
    if (N % 2 == 0) PI = 2;
    else PI = 1;
}

void estadisticas::leevector(int N) {
    nVector = new int[N];
    cout << "Simulando Estaturas:\n ";
    for (int i = 0; i < N; i++)
        nVector[i] = rand() % 36 + 150;
    //cin >> nVector[i];
};

void estadisticas::escribirvector(int N) {
    cout << "Estaturas simuladas " << endl;
    cout << '{';
    for (int i = 0; i < N; i++)
        cout << *(nVector + i) << ' ';
    cout << '}';
}

void estadisticas::ordenavector(int N) {
    int aux;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (nVector[i] < nVector[j]) {
                aux = nVector[i];
                nVector[i] = nVector[j];
                nVector[j] = aux;
            }
        }
    }
}

void estadisticas::mostrarvector(int N) {
    cout << "\nAlumnos Ordenados de menor a mayor estatura: " << endl;
    for (int i = 0; i < N; i++) {
        cout << *(nVector+i) << ' ';
    }
}

double estadisticas::Media(int N) {
    int S = 0;
    for (int i = 0; i < N; i++)
        S = S + nVector[i];
    media = S * 1.0 / N;
    return(media);
}
```

```

float estadisticas::Mediana(int N) {
    if (PI == 1) { mediana = (nVector[N / 2]) * 1.0; }
    else { mediana = (nVector[N / 2 - 1] + nVector[N / 2]) * 1.0 / 2; }
    return(mediana);
}

void estadisticas::Moda(int N) {
    nModa = new int[N]; //cuenta cuantas veces se repite cada numero
    for (int i = 0; i < N; i++) {
        nModa[i] = 0;
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (nVector[i] == nVector[j]) nModa[i] = nModa[i] + 1;
        }
    }
    //Hallando el mayor número de veces que se repite un valor
    int Mayor = 0;
    for (int i = 0; i < N; i++) {
        if (Mayor < nModa[i]) Mayor = nModa[i];
    }

    for (int i = 0; i < N; i++) {
        int v = 0;
        if (nModa[i] == Mayor) {
            for (int j = 0; j < i; j++) {
                if (nVector[i] == nVector[j]) v = 1;
            }
            if (v == 0) cout << nVector[i] << " ";
        }
    }
}

double estadisticas::DesvEstd(int N) {
    float S2 = 0, VarMuestral;
    for (int i = 0; i < N; i++)
        S2 = (S2 + powf(nVector[i], 2)) * 1.0;
    VarMuestral = (S2 - N * media * media) * 1.0 / (N - 1);
    desv_estd = powf(VarMuestral, 1.0 / 2);
    return(desv_estd);
}

void main() {
    int N;
    estadisticas mb545;
    cout << "Ingrese la cantidad de Alumnos: "; cin >> N;
    mb545.ParImpar(N);
    mb545.leevector(N);
    mb545.escribirvector(N);
    mb545.ordenavector(N);
    mb545.mostrarvector(N);
    cout << "\nLa media es: " << mb545.Media(N);
    cout << "\nLa mediana es: " << mb545.Mediana(N);
    cout << "\nLa moda es: ";
    mb545.Moda(N);
    cout << "\nLa desviacion estandar es: " << mb545.DesvEstd(N) << endl;
    system("pause");
}

//PREGUNTA 2

class serie {
protected:
    float X;

```

```

    int N;
public:
    serie();
    serie(float, int);
    double factor();
    double senx();
};
//Implementacion
serie::serie() {
    X = 54;
    N = 6;
}

serie::serie(float Y, int M) {
    X = Y;
    N = M;
}

double serie::factor() {
    return(k / pi);
}

double serie::senx() {
    double sum = 0, Ang = 0, ser;
    int fact = 1, prod;
    Ang = factor(); //X * k;
    for (int i = 1; i <= N; i++) {
        int term = i;
        for (int i = 1; i <= (2 * term - 1); i++) {
            prod = i;
            fact = fact * prod;
        }
        ser = (powf(-1, i + 1) * powf(Ang, 2 * i - 1)) / fact;
        sum = sum + ser;
    }
    return(sum);
}

class Simpson :public serie {
protected:
    float* func;
public:
    Simpson(); //constructor
    Simpson(int); //constructor
    void generav();
};
//Implementacion
Simpson::Simpson()
{
    N = 18;
}
Simpson::Simpson(int Q)
{
    N = Q;
}
void Simpson::generav()
{
    float simp;
    func = new float[N];
    for (int i = 0; i < N; i++)
        func[i] = 0.5 + i * (1.0 / 3);

    for (int i = 0; i < N; i++) {

```

```

        func[i] = powf(func[i], 1.0 / 2) - powf(func[i] - func[i] / 5, 1.0 / 3) +
func[i];
    }
    simp = (3.0 / 8) * (1.0 / 3) * (func[0] + 3 * func[1] + 3 * func[2] +
func[3]);

    cout << simp;
}

```

```

void main() {
    //Para simpson(N), el 'N' debe ser igual a 4 a más obligatoriamente
    serie d; Simpson D;
    cout << "La serie: " << d.senx() << endl;
    cout << "La funcion: "; D.generav();
    cout << endl;

    system("pause");
}

```

//PREGUNTA 3

```

void Cestrellas::OnBnClickedRadio1()
{
    CClientDC q(this);
    CPen k; CBrush l; CFont
tl;

```

```

k.CreatePen(PS_SOLID, 5,
RGB(0, 0, 255));
q.SelectObject
(k);
l.CreateSolidBru
sh(RGB(255,
250, 240));
q.SelectObject
(l);
q.SetBkMode(
TRANSPARE
NT);
q.SetTextColor(RGB(255, 0, 0));
tl.CreateFontW(30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
_T("Arial Black"));q.SelectObject(tl);
q.TextOutW(20, 10, _T("ESTRELLA DE 4 PUNTAS"));

```

C
C
l
i
e
n
t
D

```

C
p
(
t
h
i
s
)
;
C
P
e
n
t
;
CBrush f;
t.CreatePen(PS_SOLID, 5, RGB(0, 0, 255));
p.SelectObject
(t);
f.CreateSolidBru
sh(RGB(0, 255,
0));
p.SelectObject
(f);
POINT v[100];
float pi = atan(1.0) * 4; //Definimos el valor de pi
int r, ri = 50, re = 100; //Radio interno y externo de la estrella
int corini_x = 170, corini_y = 150; //Coordenadas iniciales del origen

for (int i = 0; i <= 7; i++) {
    i
    f
    (
    i
    %
    2
    =
    =
    0
    )
    r
    =
    r
    e
    ;
    e
    l
    s
    e
    r

```

}

Radio Button 2

{

```

C
p
(
t
h
i
s
)
;
C
P
e
n
t
;
CBrush f;
t.CreatePen(PS_SOLID, 5, RGB(255, 0, 0));
p.SelectObject
(t);
f.CreateSolidBru
sh(RGB(0, 0,
255));
p.SelectObject
(f);

POINT v[100];
float pi = atan(1.0) * 4; //Definimos
el valor de pi=3.1415... int r, ri =
50, re = 100; //Radio interno y
externo de la estrella
int corini_x = 170, corini_y = 150; //Coordenadas iniciales del origen

for (int i = 0; i <= 9; i++) {
    i
    f
    (
    i
    %
    2
    =
    =
    0
    )
    r
    =
    r
    i
    ;
    e
    l

```

```

s
e
r
=
r
e
;

v[i].x = corini_x + r *
cos((i * (pi / 5)) + (pi /
2));v[i].y = corini_y + r
* sin((i * (pi / 5)) + (pi /
2)); }

```

```

p.Polygon(v, 10);}

```

Radio Button 3

```

void Cestrellas::OnBnClickedRadio3()
{
    CClientDC q(this);
    CPen k; CBrush l;
    CFont tl;
    k.CreatePen(PS_SOLID, 5, RGB(0, 0,
255));
    q.SelectObject
(k);
    l.CreateSolidBrush(RGB(255,
250, 240));
    q.SelectObject
(l);
    q.SetBkMode(TRANSPARENT);
    q.SetTextColor(RGB(255, 0, 0));
    tl.CreateFontW(30, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
_T("Arial Black"));q.SelectObject(tl);
    q.TextOutW(20, 10, _T("ESTRELLA DE 6 PUNTAS"));

    CClientDC p(this);
    C
P
e
n
t
;
C
B
r
u
s

```



```

h
f
;
t.CreatePen(PS_SOLID, 5, RGB(0, 255, 0));
p.SelectObject
(t);
f.CreateSolidBru
sh(RGB(255,0 ,
0));
p.SelectObject
(f);
POINT v[100];
float pi = atan(1.0) * 4; //Definimos
el valor de pi=3.1415... int r, ri =
50, re = 100; //Radio interno y
externo de la estrella
int corini_x = 170, corini_y = 150; //Coordenadas iniciales del origen

for (int i = 0; i <= 11; i++) {
    i
    f
    (
    i
    %
    2
    =
    =
    0
    )
    r
    =
    r
    e
    ;
    e
    l
    s
    e
    r
    =
    r
    i
    ;

    v[i].x = corini_x + r *
    cos((i * (pi / 6)) + (pi /
    2)); v[i].y = corini_y + r
    * sin((i * (pi / 6)) + (pi /
    2));
}

```

```
p.Polygon(v, 12); }
```

Botón Limpiar

```
void Cestrellas::OnBnClickedButton2()
{ OnOK(); }
```

**Pru
eba
del
pro
gra
ma
Cor
rida
1**



Corrida 2



//PREGUNTA 4

Solución

```
int fibo(int n)
{
    if (n == 1) return 1;
    else if (n == 2) return 1;
    else return fibo(n - 1) + fibo(n - 2);
}
```

```

int esprimo(int n)
{
    if (n == 1) return 0;
    for (int d = 2; d < n; d++)
        if (n % d == 0) return 0;
    return 1;
}

void CMFCApplication12Dlg::OnPrincipalSeriedefibonacci()
{
    m_opcion = 0;
    UpdateData(0);
}

void CMFCApplication12Dlg::OnPrincipalSeriedeprimos()
{
    m_opcion = 1;
    UpdateData(0);
}

void CMFCApplication12Dlg::OnPrincipalEjecutar()
{
    UpdateData(true);
    CString cad; int s = 0;
    m_lis1.ResetContent();
    switch (m_opcion)
    {
    case 0: //1 1 2 3 5 8
    {
        for (int i = 1; i <= m_nt; i++)
        {
            if (m_op1 == true)
                cad.Format(_T("%d) %d"), i, fibo(i));
            else
                cad.Format(_T("%d"), fibo(i));
            m_lis1.AddString(cad);
            s = s + fibo(i);
        }

    }
    break;
    case 1:
    {
        for (int i = 1, c = 1; c <= m_nt; i++)
        {
            if (esprimo(i))
            {
                if (m_op1 == true)
                    cad.Format(_T("%d) %d"), c, i);
                else
                    cad.Format(_T("%d"), i);
                m_lis1.AddString(cad);
                c++;
                s = s + i;
            }
        }
    }
    break;
}

```

```

    }
    if (m_op2 == true)
    {
        m_lis1.AddString(_T(""));
        m_lis1.AddString(_T("Reporte final"));
        cad.Format(_T("Suma de la serie: %d"), s);
        m_lis1.AddString(cad);
    }
    UpdateData(false);
}

void CMFCApplication12Dlg::OnPrincipalSalr()
{
    OnOK();
}

void CMFCApplication12Dlg::OnPrincipalBorrarlista()
{
    m_lis1.ResetContent();
}

```