Documento anónimo

Ejercicio3.pdf

Enero 2018 - Enunciados y Soluciones PDF

1º Fundamentos de Programación

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

```
//(3 Puntos) ASCII Art
class SecuenciaCaracteresEnteros
{
    private:
        static const int TAMANIO = 100;
        int vector_privado[TAMANIO];
        int total_utilizados;
    public:
        SecuenciaCaracteresEnteros()
        {
            total_utilizados=0;
        }
        int TotalUtilizados(){  return total_utilizados;}
        int Elemento(int indice)
        {
            if (indice >=0 && indice < total_utilizados)
            {
                return vector_privado[indice];
            }
        }
        void Aniade(int nuevo)
        {
            if (total_utilizados< TAMANIO)
            {
                vector_privado[total_utilizados]=nuevo;
                total_utilizados++;
            }
        }
        void EliminaTodos()
        {
            total_utilizados=0;
        }
};
class TablaRectangularEnteros
{
```

```
private:
    static const int NUM_FILS=10;
    static const int NUM_COLS=10;
    int matriz_privada[NUM_FILS][NUM_COLS];
    int filas_utilizadas;
    int cols_utilizadas;
public:
    TablaRectangularEnteros()
    {
        filas_utilizadas=0;
        cols_utilizadas=0;
    }
    TablaRectangularEnteros(int n_cols)
    {
        filas_utilizadas=0;
        cols_utilizadas=n_cols;
    }
    int Filas_Utiles()
    {
        return filas_utilizadas;
    }
    int Cols_Utiles(){ return cols_utilizadas;}

    int Elemento(int indice_f,int indice_c)
    {
        if (indice_c >=0 && indice_f >= 0 && indice_f < filas_utilizadas && indice_c < cols_utilizadas)
            return matriz_privada[indice_f][indice_c];
    }
    SecuenciaCaracteresEnteros Fila(int indice_f)
    {
        if (indice_f >=0 && indice_f < filas_utilizadas)
        {
            SecuenciaCaracteresEnteros new1;
            for (int i = 0 ; i < cols_utilizadas ; i++)
            {
```

```
            new1.Aniade(matriz_privada[indice_f][i]);
        }
        return new1;
    }
}
void Aniade(SecuenciaCaracteres fila_nueva)
{
    if (filas_utilizadas < NUM_FILS && cols_utilizadas < fila_nueva.TotalUtilizados())
    {
        for (int i = 0 ; i < cols_utilizadas ; i++)
        {
            matriz_privada[filas_utilizadas][i]=fila_nueva.Elemento(i);
        }
    }

}
void EliminaTodo()
{
    filas_utilizadas=0;
    cols_utilizadas=0;
}
TablaRectangularCaracteres toAscii()
{
    TablaRectangularCaracteres m(cols_utilizadas);
    for(int i = 0 ; i < filas_utilizadas ; i++)
    {
        SecuenciaCaracteres cadena;
        cadena.EliminaTodos();
        for (int j = 0 ; j < cols_utilizadas ; j++)
        {
            char caracter;
            int elem = matriz_privada[i][j]%255;
            if (elem >0 && elem < 70)
            {
                caracter = '@';
```

```
                    }
                    else if(elem < 130)
                    {
                        caracter = '&';
                    }
                    else if(elem < 170)
                    {
                        caracter = ':';
                    }
                    else if(elem < 210)
                    {
                        caracter = '*';
                    }
                    else if(elem < 255)
                    {
                        caracter = ' ';
                    }
                    cadena.Aniade(caracter);

                }
                m.Aniade(cadena);
            }
            return m;
        }
};
class SecuenciaCaracteres
{
    private:
        static const int TAMANIO = 100;
        char vector_privado[TAMANIO];
        int total_utilizados;
    public:
        SecuenciaCaracteres()
        {
            total_utilizados=0;
```

```
        }
        int TotalUtilizados(){  return total_utilizados;}
        char Elemento(int indice)
        {
            if (indice >=0 && indice < total_utilizados)
            {
                return vector_privado[indice];
            }
        }
        void Aniade(int nuevo)
        {
            if (total_utilizados< TAMANIO)
            {
                vector_privado[total_utilizados]=nuevo;
                total_utilizados++;
            }
        }
        void EliminaTodos()
        {
            total_utilizados=0;
        }
};
class TablaRectangularCaracteres
{
    private:
        static const int NUM_FILS=10;
        static const int NUM_COLS=10;
        char matriz_privada[NUM_FILS][NUM_COLS];
        int filas_utilizadas;
        int cols_utilizadas;
    public:
        TablaRectangularCaracteres()
        {
            filas_utilizadas=0;
            cols_utilizadas=0;
```

```
}
TablaRectangularCaracteres(int n_cols)
{
    filas_utilizadas=0;
    cols_utilizadas=n_cols;
}
int Filas_Utiles()
{
    return filas_utilizadas;
}
int Cols_Utiles(){ return cols_utilizadas;}

char Elemento(int indice_f,int indice_c)
{
    if (indice_c >=0 && indice_f >= 0 && indice_f < filas_utilizadas && indice_c < cols_utilizadas)
        return matriz_privada[indice_f][indice_c];
}
SecuenciaCaracteres Fila(int indice_f)
{
    if (indice_f >=0 && indice_f < filas_utilizadas)
    {
        SecuenciaCaracteres new1;
        for (int i = 0 ; i < cols_utilizadas ; i++)
        {
            new1.Aniade(matriz_privada[indice_f][i]);
        }
        return new1;
    }
}
void Aniade(SecuenciaCaracteres fila_nueva)
{
    if (filas_utilizadas < NUM_FILS && cols_utilizadas < fila_nueva.TotalUtilizados())
    {
        for (int i = 0 ; i < cols_utilizadas ; i++)
        {
```

```
                    matriz_privada[filas_utilizadas][i]=fila_nueva.Elemento(i);
                }

            }
        }
        void EliminaTodo()
        {
            filas_utilizadas=0;
            cols_utilizadas=0;
        }

};
```