

Vectores y matrices

Fundamentos de Programación
Fundamentos de Programación I

Ejemplo de utilización de vectores: gráfico de producción

En una compañía nos han encargado escribir un programita que despliegue un gráfico de barras dónde se muestra la diferencia de producción con respecto al máximo de producción por planta. Existen en la compañía 10 plantas diferentes. La salida debe ser un gráfico similar al siguiente:

```
Planta #1 (6000): *****
Planta #2 (0)
Planta #3 (7000): *****
.
.
Planta #10:**
```

Cada * representa 1000 unidades de producción.

Entrada: Hay 10 plantas diferentes de producción por lo tanto el usuario introducirá la producción para cada planta (10 números diferentes).

Salida: El gráfico con la siguiente información:

```
Planta n° (max-produccion): ****
```

```

/*****
Programa que muestra un gráfico de barras de producción.
Fecha: Diciembre 2003
Autores:...
*****/

#include <iostream.h>

//Definicion de constantes

const int MAX_PLANTAS = 10;
const int ESCALA = 100;

//Defincion de tipos

typedef int PlantProd[MAX_PLANTAS];

//Prototipos de funciones
void Leer_Datos(PlantProd plantas);
int Calcular_Maximo(const PlantProd plantas);
void Representar_Grafica(const PlantProd plantas, int max);

```

```

//Funcion principal
int main()
{
    PlantProd plantas;
    int maximo;

    //Presentacion del programa
    cout << "Este programa dibuja un grafico de barras de la diferencia de..\n";

    Leer_Datos(plantas);
    maximo=Calcular_Maximo(plantas);
    Representar_Grafica(plantas,maximo);

    Return 1;
}

void Leer_Datos(PlantProd plantas)
{
    int i;

    cout << "Introduce las unidades de producción para las 10 plantas\n";
    for (i = 0 ; i < MAX_PLANTAS ; i++)
    {
        cout << "Planta #" << i << endl;
        cin >> plantas[i];
    }
    return ;
}

```

```

int Calcular_Maximo(const PlantProd plantas)
{
    int max;
    int i;

    max = plantas[0];
    for (i = 1 ; i < MAX_PLANTAS ; i++)
    {
        if ( max < plantas[i])
            max = plantas[i];

    }
    return max;
}

```

```

void Representar_Grafica(const PlantProd plantas, int max)
{
    //Declaración de variables
    int i,j;
    int asteriscos;
    char ch;

    for ( i = 0; i < MAX_PLANTAS; i++)
    {
        cout << "Planta #"<< i ;
        asteriscos = (max - plantas[i]) / ESCALA;
        for ( j = 0; j < asteriscos ; j++)
            cout << "*";
        cout << endl;
    }
    return ;
}

```

Barajar un mazo de cartas

Imaginemos que estamos desarrollando un juego de cartas (ejemplo solitario). Para ello es necesario barajar las cartas para cada nueva partida. Vamos a implementar una función que dado un mazo de cartas lo baraje.

La estructura de datos que vamos a utilizar para representar la baraja será un vector de enteros. Un determinado número entero representará una determinada carta de la baraja española según lo siguiente:

Oros [0..9]

Copas [10..19]

Espadas [20..29]

Bastos [30..39]

Se pide realizar una función que dado un mazo con las cartas en una posición determinada, las mueva de posición de forma aleatoria.

```
#include <iostream.h>
#include <math.h>
#include <stdlib.h>
const int TAM=40;
typedef int Baraja[TAM];
//prototipo de funciones
void barajar(Baraja mazo, int tam);
void iniciar_baraja(Baraja mazo, int tam) ;
void imprimir_vector(const Baraja mazo, int tam);
int main()
{
    Baraja mimazo;
    int tam=TAM;
    float nada;
    char ch;
    //presentacion programa
    cout << "Este programa baraja un mazo de cartas aleatoriamente\n";
    //INiciar baraja para que este ordenada
    iniciar_baraja(mimazo, tam);
    nada=M_PI*2;
    cout << "Baraja ordenada\n";
    imprimir_vector(mimazo, tam);
    barajar(mimazo, tam);
    cout << "\nYa estan las cartas barajadas\n";
    imprimir_vector(mimazo, tam);
    system("pause");
    return 0;
}
```

Prototipo de
funciones



```
void iniciar_baraja(Baraja mazo, int tam)
{
    int i;

    for(i = 0 ; i < tam ; i++)
    {
        mazo[i]=i;
    }
}
```

```
void barajar(Baraja mazo, int tam)
{
    int i, tmp;
    int indice, desp;
    //Almaceno una nueva carta en cada posicion
    for( i = 0; i < tam ; i++)
    {
        //Genero un numero aleatorio
        //Corresponde al indice por el que sustituir
        desp=i;
        indice = int(rand()/(float)RAND_MAX * (TAM -i)) + desp;
        //Intercambio valores de variables
        tmp=mazo[i];
        mazo[i]=mazo[indice];
        mazo[indice]=tmp;
    }
    return ;
}
```

```
void imprimir_vector(const Baraja mimazo, int tam)
{
    int i;

    for (i = 0; i < tam ; i++)
    {
        if ( (i % 5 )== 0 )
            cout << endl;
        cout << mimazo[i] << ' ';
    }
    cout << endl;
    return;
}
```

Eliminación de duplicados

Imaginemos que estamos haciendo una encuesta porque estamos interesados en saber la edad de los alumnos que asisten a una clase.

Hemos pensado que sería útil tener un programa que dadas las edades de los alumnos las resumiera y las imprimiera pero sin repeticiones.

El programa que queremos desarrollar debe pedir la edad de todos los alumnos de la clase, sin ningún tipo de orden y después imprimir las edades pero sin que aparezcan valores repetidos.

```

#include <iostream.h>
#include <stdlib.h>

const int MAX=100;
typedef int Edades[MAX];

//PROTOTIPOS

int leer_vector(Edades mis_Edades);
void imprimir_vector(Edades mis_Edades, int tam);
void eliminar_duplicados(Edades mis_Edades, int &tam);

int main()
{
    Edades ed_clase;
    int tam=0;
    //Presentacion programa

    cout << "Introduce las Edades de cada una de las personas de la clase" << endl;
    cout << "Para finalizar introduce un numero negativo\n";

    //Lectura de datos (vector vacio y devuelve el tamaño
    tam=leer_vector(ed_clase);

    eliminar_duplicados(ed_clase, tam);

    imprimir_vector(ed_clase, tam);

    system("PAUSE");
    return 0;
}

```

```

int leer_vector(Edades mis_Edades)
{
    int i;
    int num;
    cin >> num;
    i=0;
    //Bucle de repetición
    //Para si el numero es negativo
    while(num > 0 && i< MAX)
    {
        mis_Edades[i]=num;
        cin >> num;
        i++;
    }
    //Devuelvo el tamaño del vector
    return i;
}

```

```

//Funcion para imprimir el contenido del vector
void imprimir_vector(Edades mis_Edades, int tam)
{
    int i;

    for (i = 0; i < tam ; i++)
        cout << mis_Edades[i] << ' ';

    cout << endl;
}

```

```

//Funcion que elimina los valores repetidos en un vector pasado por referencia
void eliminar_duplicados(Edades mis_Edades, int& tam)
{
    int i, j,k;
    //Recorro el vector varias veces
    for (i = 0; i< tam ; i++)
        for (j = i+1; j < tam; j++)
            if( mis_Edades[i]== mis_Edades[j])
            {
                //Quito ese elemento del vector y muevo el resto
                for ( k = j  ; k < tam ; k++)
                {
                    mis_Edades[k]=mis_Edades[k+1];
                }
                j = j - 1;
                tam = tam - 1;
                cout << "Vector ahora es:\n indices: " << i << ' ' << j << endl;
                imprimir_vector(mis_Edades,tam);
            }
    return ;
}

```

Calculo del histograma de los valores almacenados en una matriz

Implementa una programa que permita calcular el histograma de los distintos valores almacenados en una matriz de tamaño TAMxTAM.

El programa pedirá al usuario que introduzca el tamaño de la matriz y los valores que se almacenarán en ella, calculará el histograma y lo imprimirá por pantalla.

```

#include <iostream.h>

//Definición de constantes
const int TAM = 100;
const int NIVELES = 40;

//Definicion de tipos
typedef int Matriz[TAM][TAM];
typedef int Histograma[NIVELES];

//Prototipos de funciones
void leer_matriz(Matriz mat, int& filas, int& column);
void imprimir_matriz(const Matriz mat, int filas, int column);
void calcular_histograma(const Matriz mat, int filas, int column, Histograma hist, int niveles);
void imprimir_histograma(const Histograma hist, int niveles);

int main()
{
    Matriz mat;
    Histograma hist;
    int fil,col;
    int niveles;

    niveles=NIVELES;

    //Leer datos
    leer_matriz(mat,fil,col);

    //Imprimir matriz
    imprimir_matriz(mat,fil,col);

    //Calculo del histograma
    calcular_histograma(mat,fil,col,hist,niveles);
    //Imprimo histograma

    imprimir_histograma(hist,niveles);
    return 0;
}

```

```

void leer_matriz(Matriz mat, int& filas, int& column)
{
    int i,j;

    cout << "Introduce el tamaño de la matriz (filas columnas):\n";
    cin >> filas;
    cin >> column;

    for(i = 0; i < filas; i++)
        for (j = 0; j < column ; j++)
        {

            cout << "Elemento :" << i << ' ' << j << endl;
            cin >> mat[i][j];

        }

    return ;
}

```

```

void imprimir_matriz(const Matriz mat, int filas, int column)
{
    int i , j;

    for(i = 0; i < filas; i++)
    {
        for (j = 0; j < column ; j++)
        {
            cout << mat[i][j] << ' ';

        }
        cout << endl;
    }

    return ;
}

```

```

//Funcio para calcular el histograma de una imagen
void calcular_histograma(const Matriz mat,int filas,int column,Histograma hist, int niveles)
{
    int i ,j;
    int valor;

    //Inicializo los contadores
    for ( i = 0; i< niveles; i++)
        hist[i]=0;

    //Recorro la matriz para contar valores repetidos
    for (i = 0; i < filas ; i++)
        for( j = 0; j < column ; j++)
        {
            valor = mat[i][j];
            if ( valor < niveles)
                hist[valor] ++;

        }
    return;
}

//Imprimo por pantalla el histograma
void imprimir_histograma(const Histograma hist, int niveles)
{
    int i;

    for (i = 0; i < niveles ; i++)
        cout << hist[i] << endl;

    return;
}

```