Vectors i matrius

- Els vectors agrupen variables de un mateix tipus amb un nom únic.

Tipo Nombre[NumElementos] → en C++ → *int vector[10];*

- Tipo indica el tipus del elements del vector.
- El tipus pot ser cuansevol tipus simple o estructurat.
  - Ex: vectors de enters, de reals, de vectors (matrius), d'alumnes, …

- NumElementos es el tamany del vector (nombre d'elements) i ha de ser sempre una constant, per exemple:.

```
const int MAX_NUMEROS = 10;
typedef int Vector10[MAX_NUMEROS];
Vector10 mi_vector;
```

Vector correctament definit
(tamany com a constant,
Vector com a nou tipus
Variable del tipus recent definit)

- Així tenim 10 enters agrupats i es podran tractar com una unica variable (mi_vector).

Indexació → acces als elements mitjançant una variable index.

```
int aux, indice = 5;
aux = mi_vector[indice];   // aux conte el element 5 del vector
```

Generalment gastarem bucles for per a recorrer tots els elements del vector, donat que sempre sabem les iteracions necessaries.

Per exemple, inicialment tots els elements del vector valen 0:

```
for(int i = 0; i < 10; i++)
    mi_vector[i] = 0;
```

… si volem escriure per pantalla un vector:

```
for(int i = 0; i < MAX_NUMEROS; i++)
    cout << mi_vector[i];
```

Paso de arrays a funcions → arrays com paràmetres

No es posible pasar vectors per valor, es a dir, si pasem un array a una funció, estem pasan-lo per referencia encara que no tinga &.

→ Per que? .. Hi han varies raons … eficiencia, representació dels vectors en memoria

Per tant, podem llegir un vector per teclat d'aquesta manera:

```
void LeerVector(Vector10  v)
{
   int i;
   for(i = 0; i < MAX_NUMEROS; i++)
      cin >> v[i];
   return;
}
```

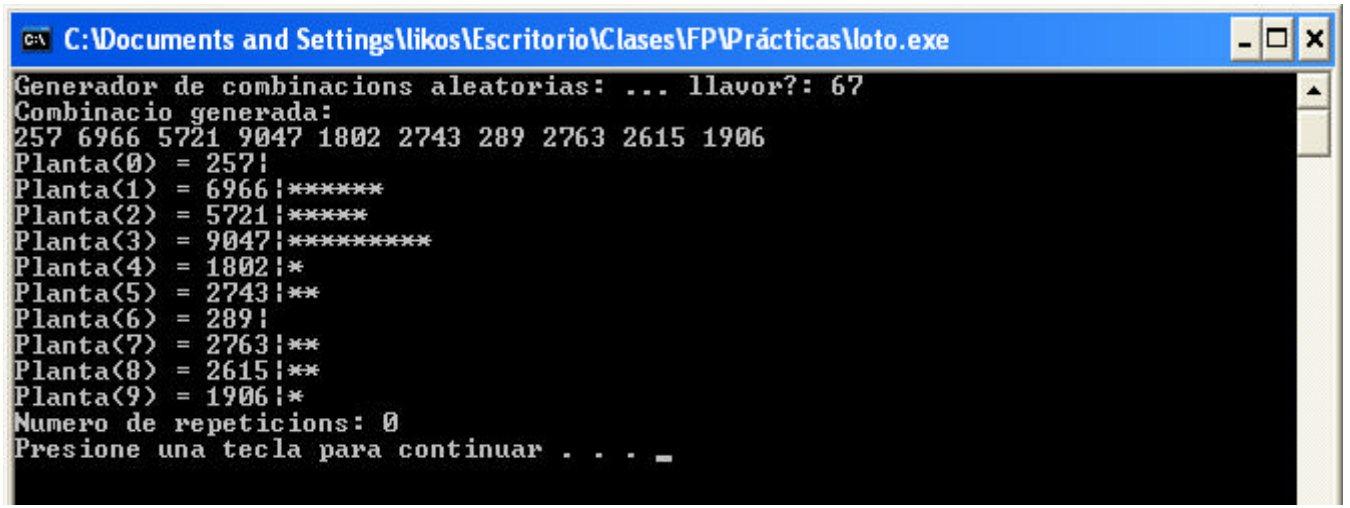
```
void MostrarVector(const Vector v)
{
int i;
for(i = 0; i < MAX; i++)
    cout << v[i];
return;
}
```

Podem fer v[0] = 1; ???
→ error de compilació.

Paràmetres **constants**. Son paràmetres que es declaren com constants, de manera que no poden ser modificats en la funció.

Per últim, una funció no deu torar valors array, es a dir, no hi ha que declarar funcions de tipus array. Si volem tonar un array, harem de fer us d'alun paràmetre

Escriure un programa que mostre un senzill gràfic de barres d'una planta de producció similar a aquest:

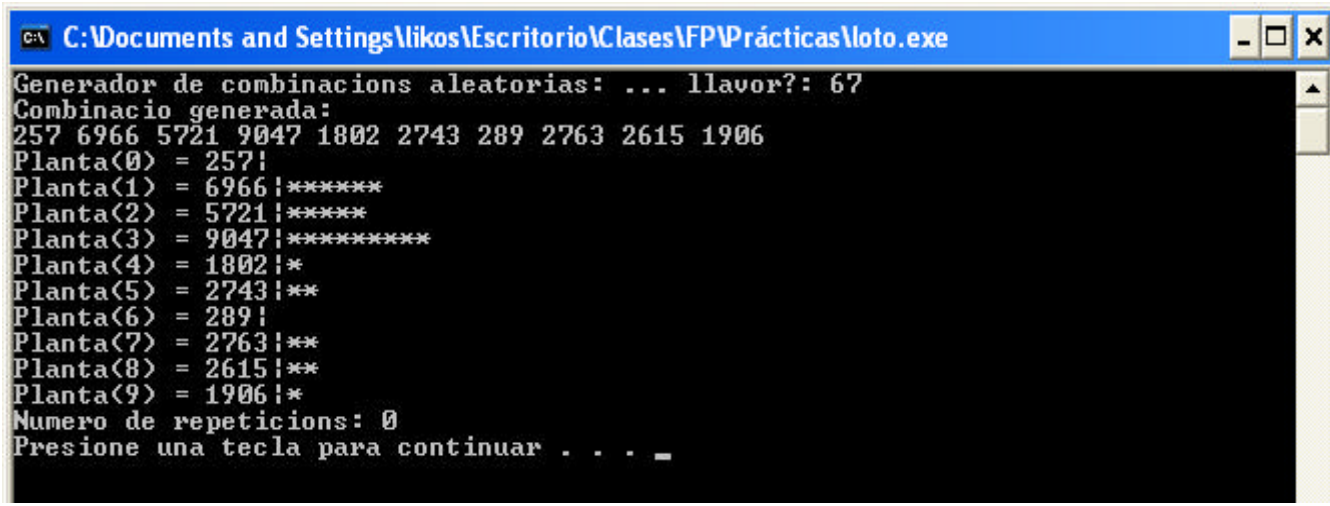


.. on cada * representa 1000 unitats de producció (Màxima producció = 10.000 → 10 asteriscos. Donat que hi han 10 plantes diferents l'usuari introduirà la producció de cada planta (10 numeros diferents). Fer una funció que torne el valor màxim.

```
#include <iostream.h>

//Definicio de constants i tipus
const int MAX_PLANTAS = 10;
const int ESCALA = 1000;
typedef  int PlantProd[MAX_PLANTAS];

 //Prototips de les funcions
void Leer_Datos(PlantProd plantas);
void Representar_Grafica(const PlantProd plantas);
```

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

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

  Leer_Datos(plantas);
  Representar_Grafica(plantas);

  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 ;
}
```

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

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

}
```

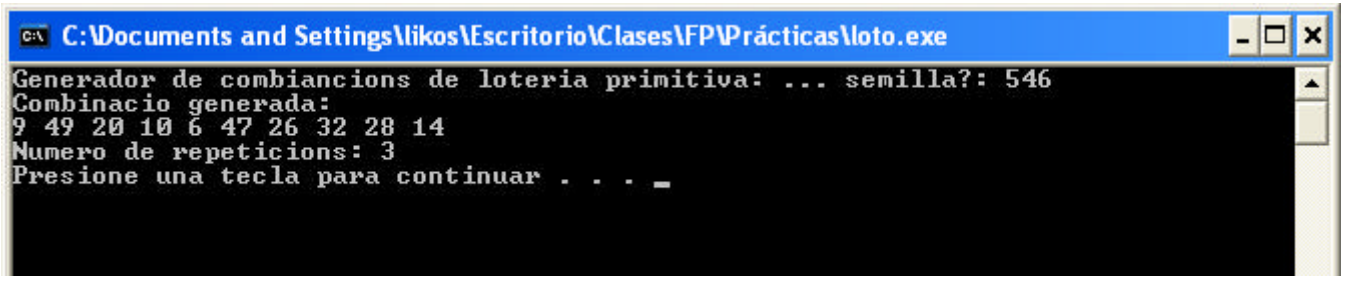


Escriure una funció més per calcular el màxim de la planta …

int Calcular_Maximo(const PlantProd plantas);

```
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;
}
```

- Exercici 14: Escriure un programa que genere combinacions de numeros (p.ej secuencies de 10 numeros) sense repeticions (considerar un rang entre 1 i 49).



- Definir els tipus de dades necessàries … i les constants
- Prototips … cuantes funcions ens faran falta?
- Definir el bucle de control principal
  (generar N numeros sense repeticions)

podem contar les repeticions que es produiexen?

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

using namespace std;

//Definicio de constants i tipus
const int MAX_LOTO = 49;
const int MAX_NUMEROS = 10;
typedef  int Combinacio[MAX_NUMEROS];


//Prototips
void imprimir(Combinacio v);
bool pertence(int num, Combinacio v);
int aleatori(int ini, int fin);
/*
 * Programa Principal
 */
int main()
{
    Combinacio c;
    int semilla, candidat, dins = 0, repe = 0;

    cout << "Generador de combinacions aleatorias: ... llavor?: " ;
    cin >> semilla;
    srand(semilla);

    do
    {
        candidat = aleatori(1, MAX_LOTO);
        if (!pertence(candidat, c))
        {
            c[dins] = candidat;
            dins++;
        }
        else
         repe++;
    } while (dins < MAX_NUMEROS);

    imprimir(c);
    cout << "Numero de repeticions: " << repe << endl;

    system("pause");
}
```

```
int aleatori(int ini, int fin)
{
   int aleat;
   aleat = ini + rand() % int(fin - ini + 1); // para incluir fin
   return aleat;
}


bool pertence(int num, Combinacio v)
{
   for(int i = 0; i < MAX_NUMEROS; i++)
    if(v[i] == num)
      return true;

   return false;
}

void imprimir(Combinacio v)
{
   cout << "Combinacio generada: " << endl;
   for(int i = 0; i < MAX_NUMEROS; i++)
     ccout << v[i] << " " ;

   cout << endl;
}
```

Càlcul del histograma d'una matriu

Fer un programa que permita calcular l'histograma dels valors emmagatzemats en una matriu de grandària TAMxTAM.

L'usuari introduirà la grandària i els valors corresponent, el programa calcuarà l'histograma i ho imprimirá per pantalla.

```
#include <iostream.h>

//Definición de constantes
const int TAM = 800;
const int NIVELES = 256;

//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 y cuento las apariciones de cada color
   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;
}
```

La baralla de cartes

Fer una funció que donat una baralla espanyola de cartes (Oros, Copas Espadas, Bastos) la inicialice aleatoriament (barallar)

La baralla serà un vector de sencers, donat que cada numero representarà una carta de la baralla española :

Oros [0..9] Copas [10..19] Espadas [20..29] Bastos [30..39]

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

const int TAM = 40;
typedef int Baraja[TAM];

//prototipo de funciones
void barajar(Baraja  mazo);
void iniciar_baraja(Baraja mazo) ;
void imprimir_vector(const Baraja mazo);

int main()
{
    Baraja mimazo;

    cout << "Este programa baraja un mazo de cartas aleatoriamente\n";
    iniciar_baraja(mimazo);
    cout << "Baraja ordenada\n";
    imprimir_vector(mimazo);
    barajar(mimazo);
    cout << "\nYa estan las cartas barajadas\n";
    imprimir_vector(mimazo);
    system("pause");
    return 0;
}
```

Prototipo de funciones

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

    for(i = 0 ; i < TAM ; i++)
       mazo[i] = i;
}



void barajar(Baraja maz)
{
   int i, tmp, indice;

   //Intercambio una nueva carta en cada posicion
   for(i = 0; i < TAM ; i++)
   {
     //Genero un numero aleatorio en el rango restante
     indice = int(rand()/(float)RAND_MAX * (TAM - i)) + i;

     //Intercambio valores de variables
     tmp = mazo[i];
     mazo[i] = mazo[indice];
     mazo[indice] = tmp;
   }
   return ;
}
```

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

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

Eliminació de duplicats

Eliminar els duplicats d'un vector d'edats (sencers) llegit per teclat …

```
#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 = 0;
     int num;

     //podem exirir amb -1
     do {
          cin >> num;
          mis_Edades[i] = num;
          i++;
     } while(num > 0 && i < MAX);

     //tornem la grandaria
     return i;
}
```

```
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;

     // per cada element, revisarem la resta del vector i buscarem repeticions
     for (i = 0; i< tam ; i++)
         for (j = i+1; j < tam; j++)
             if( mis_Edades[i]== mis_Edades[j])
             {
                 //Lleve la repetició desplaçant la resta del vector
                 for ( k = j  ; k < tam ; k++)
                     mis_Edades[k] = mis_Edades[k+1];

                 j = j - 1;  // prepare j per a la seguent iteració
                 tam = tam - 1;
              }
     return ;
}
```