Lecture 9 examples

December 10, 2018

1 2D static arrays

Declaring a 2D static array works similar as working with 1D static arrays.

Declaration needs: * type * name - identifier like any other variable * size_1 and size_2 - **MUST** be known at compilation time

- Storage is continous in memory - see slides
- Occupies size 1 x size 2 x sizeof(type) B
- element acces with double square brackets: [][]

int tabA[2][3]; - declars an array of 6 int double tabB[5][3]; - an array of 15 doubles

Declare a 2D array of ints and fill it with data using [][], than print some

In [36]:
```c
#include <stdio.h>

int main()
{
    int a[4][3];

    a[0][0] = 0;
    a[0][1] = 1;
    a[0][2] = 2;

    a[1][0] = 10;
    a[1][1] = 11;
    a[1][2] = 12;

    a[2][0] = 20;
    a[2][1] = 21;
    a[2][2] = 22;

    a[3][0] = 30;
    a[3][1] = 31;
    a[3][2] = 32;

    printf("%d %d %d", a[2][0], a[2][1], a[2][2]);
}
```

```
20 21 22
```

Print addresses of first elements in each row, what is the distance between them?

In [13]:
```c
#include <stdio.h>

int main()
{
    int a[4][3];

    printf("%p %p %p", &a[0][0], &a[1][0], &a[2][0]);
}
```

```
0x7fff8e2cba10 0x7fff8e2cba1c 0x7fff8e2cba28
```

Conclusion: storage is continous and elements are separated by row size
Can a 2D array be treated as a 1D array. Yes, since all storage is continous!

In [37]:
```c
#include <stdio.h>

int main()
{
    int a[4][3];
    int *p = &a[0][0];

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

    printf("%d %d %d", a[1][0], a[1][1], a[1][2]);
}
```

```
3 4 5
```

Can the 1D array be treated as a 2D? Yes, and no. There is a but. So the size of a row would have to be known.

Whan writing functions working with 2D arrays we need to remembar to provide the second dimension!

In [38]:
```c
#include <stdio.h>

void fill(int A[][10], int r, int c)
{
    for(int i=0; i<r; ++i)
    {
       for(int j=0; j<c; ++j)
       {
           A[i][j] = i + j + 1;
       }
    }
```

```
}
void print(int A[][10], int r, int c)
{
    for(int i=0; i<r; ++i)
    {
       for(int j=0; j<c; ++j)
       {
           printf("%d ", A[i][j]);
       }
       printf("\n");
    }
}

int main(){
    int tab[10][10];
    fill(tab, 3, 3);
    print(tab, 3, 3);
}
```

```
1 2 3
2 3 4
3 4 5
```

A different way to initialize arrays:
1D

```
In [26]: #include <stdio.h>
         int main(){
             int a[] = {1,2,3};
             printf("%d %d %d", a[0], a[1], a[2]);
         }
```

2D is also possible, but as with functions the size of a row (number of collumns must be known).

```
In [39]: #include <stdio.h>
         int main(){
             int a[][2] = {1,2,3,4};
             printf("%d %d %d %d", a[0][0], a[0][1], a[1][0], a[1][1]);
         }
```

```
1 2 3 4
```

1.0.1 An example

- Write a program illustrating workings of a 2D static array
- Add initialization function
- Distinguish the maximum size of an array, and the one used by the program

- Illustrate how to write functions with 2D arrays
- Add a function printing a 2D array
- Add a function coping to a 1D vector the diagonal from a square matrix
- Write a function coping a row, column from a 2D array
- Write a function inserting a row column into a 2D array

In [ ]:
```c
#include <stdio.h>
#define MAX_SIZE 10

void fill(int A[][MAX_SIZE], int r, int c)
{
    for(int i=0; i<r; ++i)
    {
       for(int j=0; j<c; ++j)
       {
           A[i][j] = i + j + 1;
       }
    }
}
void print(int A[][MAX_SIZE], int r, int c)
{
    for(int i=0; i<r; ++i)
    {
       for(int j=0; j<c; ++j)
       {
           printf("%d ", A[i][j]);
       }
       printf("\n");
    }
}

void copydiag(int A[][MAX_SIZE], int r, int c, int d[])
{
    for(int i=0; i<r; ++i)
    {
        d[i] = A[i][i];
    }
}
void copyrow(int A[][MAX_SIZE], int r, int ri, int d[])
{
    for(int i=0; i<r; ++i)
    {
        d[i] = A[ri][i];
    }
}
void insertrow(int A[][MAX_SIZE], int r, int ri, int d[])
{
    for(int i=0; i<r; ++i)
```

```
    {
        A[ri][i] = d[i];
    }
}

int main(){
    int tab[MAX_SIZE][MAX_SIZE];
    fill(tab, 3, 3);
    print(tab, 3, 3);
}
```