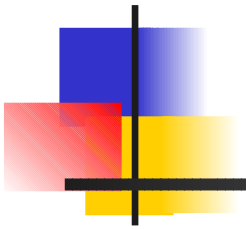
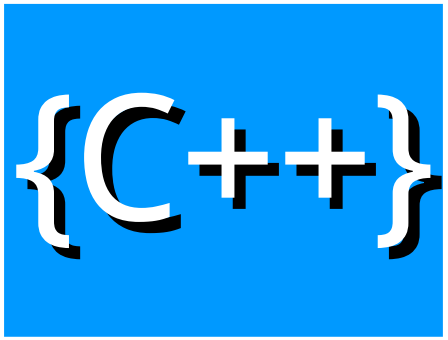
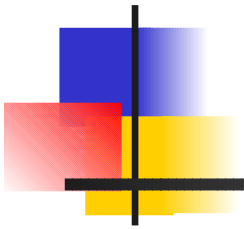


Pointers

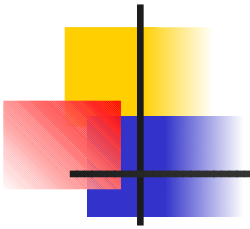
week 9



Yang-Cheng Chang
Yuan-Ze University
yczhang@g.yzu.edu.tw



Dynamic memory allocation

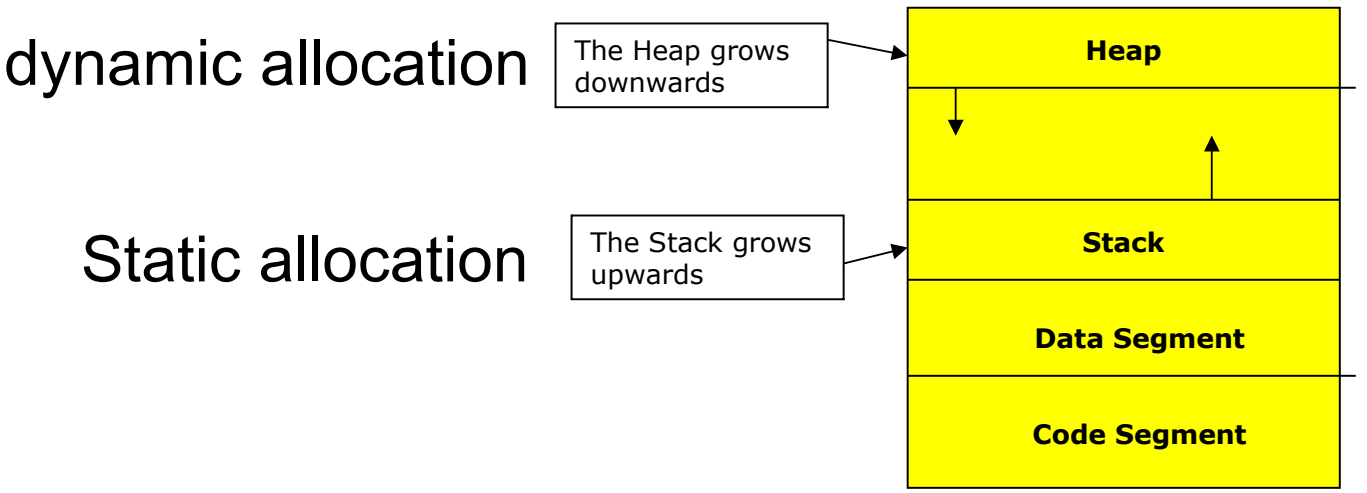


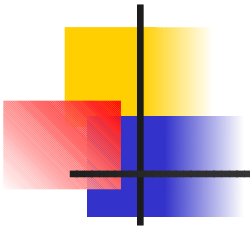
Type of Program Data

- Static Data
  —Memory allocation exists throughout execution of program
- Automatic Data
  —Automatically created at function entry, resides in activation frame of the function, and is destroyed when returning from function
- Dynamic Data
  —Explicitly allocated and deallocated during program execution by C++ instructions written by programmer

Allocation of Memory

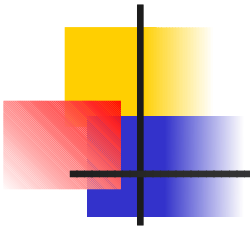
- Static Allocation
  —Allocation of memory space at compile time.
- Dynamic Allocation
  —Allocation of memory space at run time.





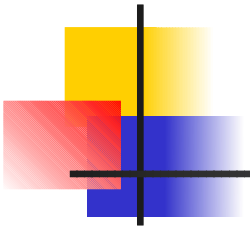
Dynamic memory allocation

- Dynamic allocation is useful when
  - —arrays need to be created whose extent is not known until run time
  - —complex structures of unknown size and/or shape need to be constructed as the program runs
  - —objects need to be created and the constructor arguments are not known until run time



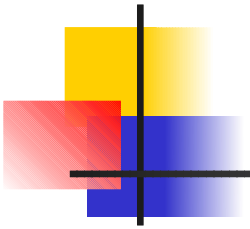
Dynamic memory allocation

- Pointers need to be used for dynamic allocation of memory
- Use the operator new to dynamically allocate space
- Use the operator delete to later free this space



The new operator

- If memory is available, the new operator allocates memory space for the requested object/array, and returns a pointer to (address of) the memory allocated.
- If sufficient memory is not available, the new operator returns NULL.
- The dynamically allocated object/array exists until the delete operator destroys it.

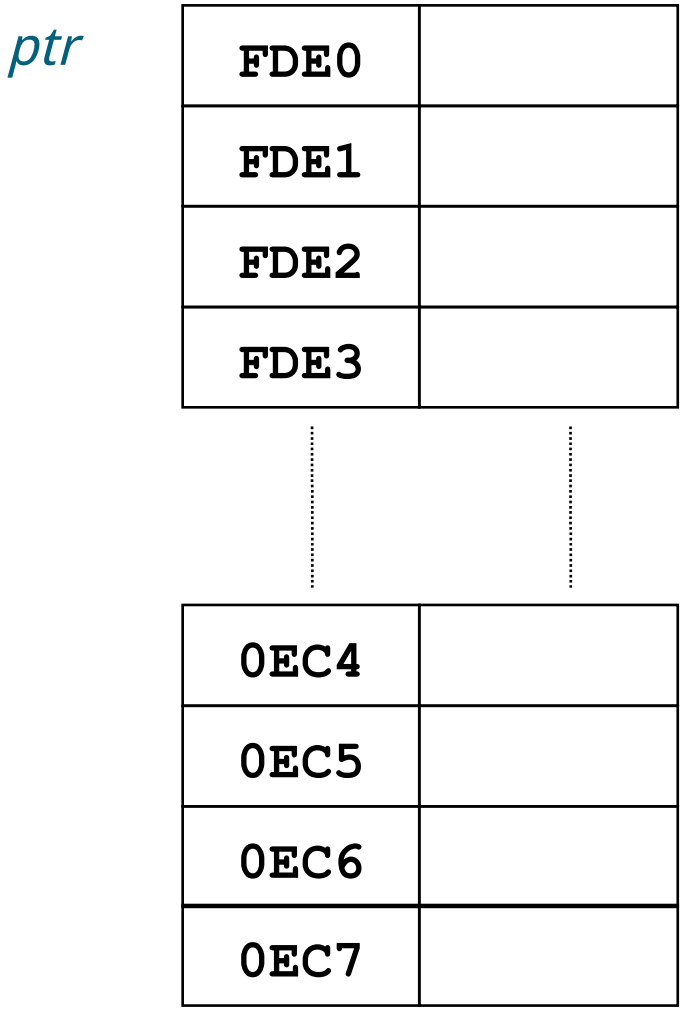


The delete operator

- The delete operator deallocates the object or array currently pointed to by the pointer which was previously allocated at run-time by the new operator.
  - —the freed memory space is returned to Heap
  - —the pointer is then considered unassigned
- If the value of the pointer is NULL there is no effect.

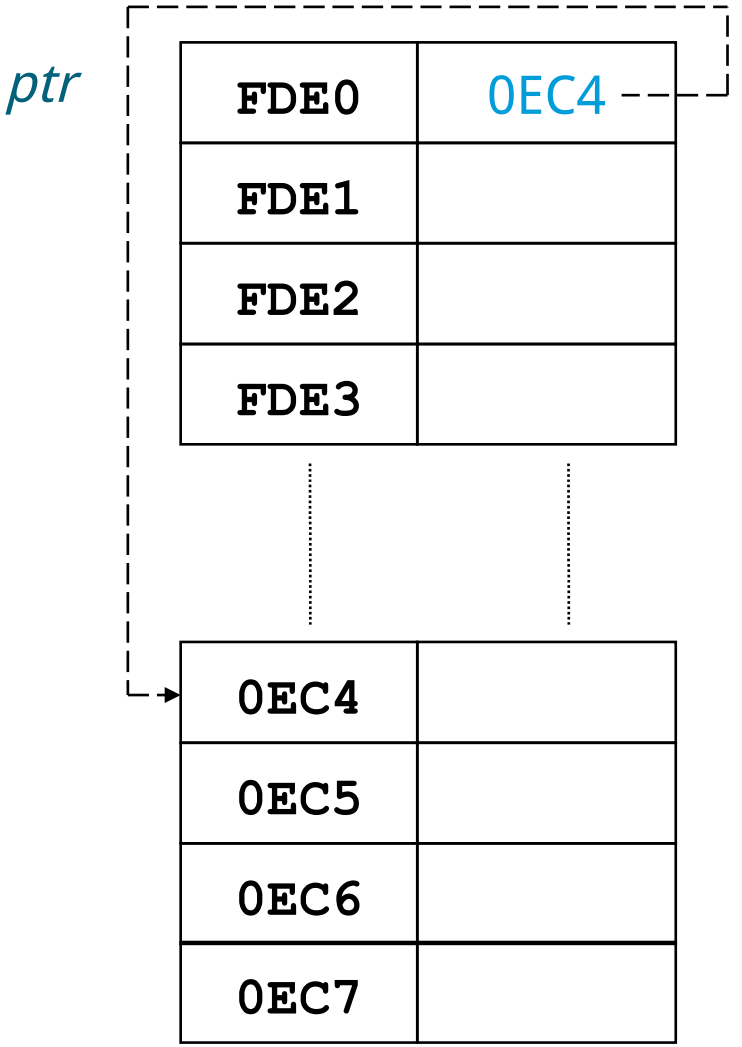
Example

```
int *ptr;
ptr = new int;
*ptr = 22;
cout << *ptr << endl;
delete ptr;
ptr = NULL;
```



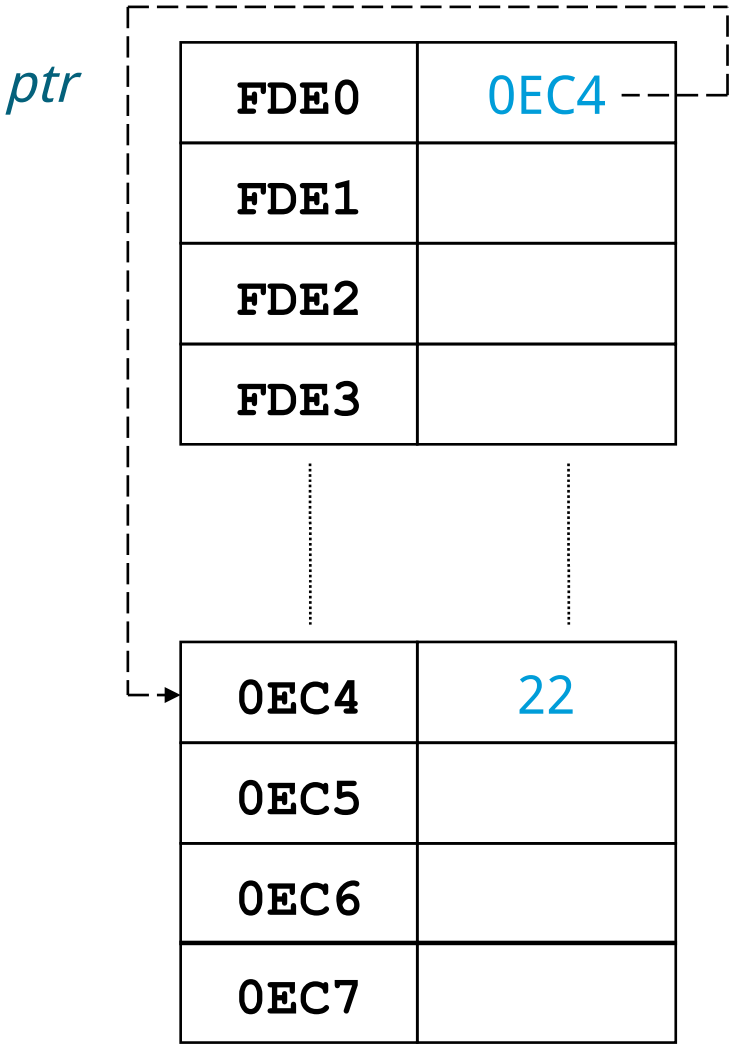
Example

```
int *ptr;
ptr = new int;
*ptr = 22;
cout << *ptr << endl;
delete ptr;
ptr = NULL;
```



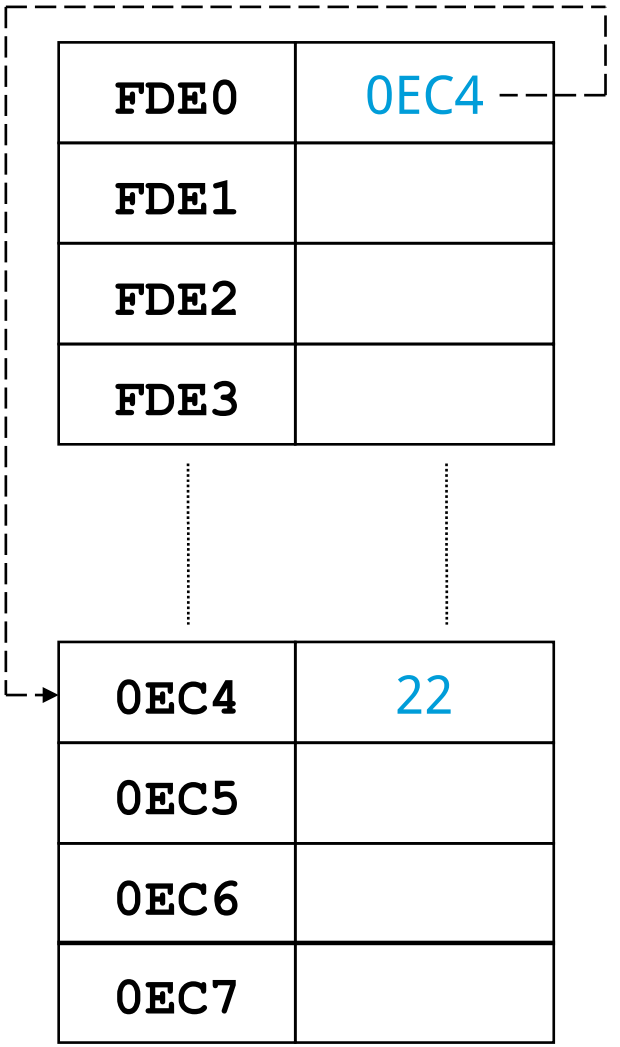
Example

```
int *ptr;
ptr = new int;
*ptr = 22;
cout << *ptr << endl;
delete ptr;
ptr = NULL;
```



Example

```
int *ptr;
ptr = new int;
*ptr = 22;
cout << *ptr << endl;
delete ptr;
ptr = NULL;
```



Output:

22

Example

```
int *ptr;
ptr = new int;
*ptr = 22;
cout << *ptr << endl;
delete ptr;
ptr = NULL;
```

ptr

FDE0	?
FDE1	
FDE2	
FDE3	

0EC4	
0EC5	
0EC6	
0EC7	

Example

```
int *ptr;
ptr = new int;
*ptr = 22;
cout << *ptr << endl;
delete ptr;
ptr = NULL;
```

ptr

FDE0	0
FDE1	
FDE2	
FDE3	

0EC4	
0EC5	
0EC6	
0EC7	

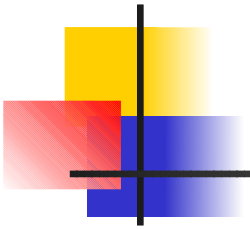
Example of dynamic array allocation

```
int* grades = NULL;
int numberOfGrades;

cout << "Enter the number of grades: ";
cin >> numberOfGrades;
grades = new int[numberOfGrades];

for (int i = 0; i < numberOfGrades; i++)
   cin >> grades[i];

for (int j = 0; j < numberOfGrades; j++)
      cout << *(grades+j) << " ";

delete [] grades;
grades = NULL;
```

Assignment 9

- Write a program to manipulate two dimension matrix. Assume the numbers of rows and columns are all 3.
- Your program must be satisfied with these conditions
  —Follow the prototype

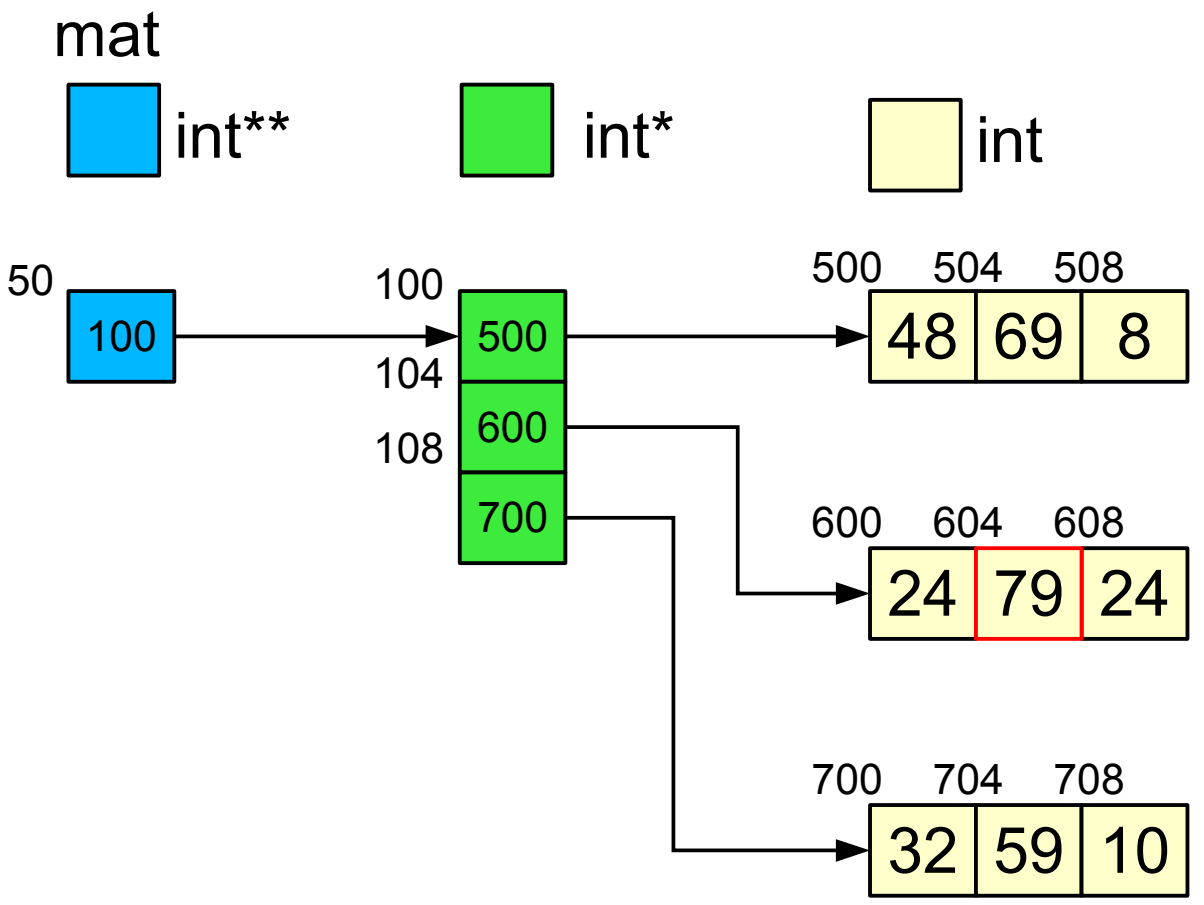
```
int createMatrixFromArray(int** &, int*);
int** multiplicateMatrix(int**, int**);
int destroyMatrix(int**);
int printMatrix(int**);
```



int createMatrixFromArray(int** &, int*);

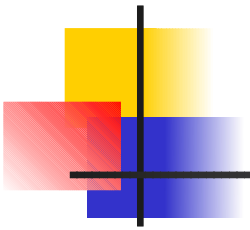
- Create two dimension array from one dimension array
  - —Using dynamic memory allocation (new) to create a two dimension array
  - —Initiate the values of this array from a given one dimension array

Using dynamic memory allocation to create two dimension array



((mat+1)+1) = 79

mat[1][1] = 79



int** multiplicateMatrix(const int** mat1, const int** mat2);

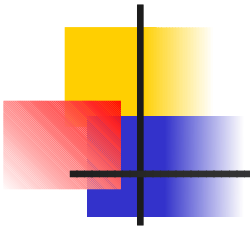
- Write a function to perform the multiplication of two matrix
  - —Allocate a new two dimension array to store the result
  - —Return the pointer of pointer which points to the result

The multiplication of two matrix

$$\begin{bmatrix} 48 & 69 & 8 \\ 24 & 79 & 24 \\ 32 & 59 & 10 \end{bmatrix} \times \begin{bmatrix} 18 & 26 & 47 \\ 90 & 31 & 36 \\ 2 & 47 & 98 \end{bmatrix} = \begin{bmatrix} 7090 & 3763 & 5524 \\ 7590 & 4201 & 6324 \\ 5906 & 3131 & 4608 \end{bmatrix}$$

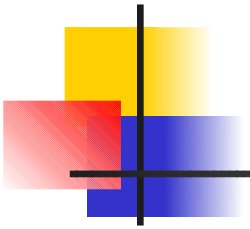
$$7090 = \begin{bmatrix} 48 & 69 & 8 \end{bmatrix} \times \begin{bmatrix} 18 \\ 90 \\ 2 \end{bmatrix} = 48 \times 18 + 69 \times 90 + 8 \times 2$$

$$4201 = \begin{bmatrix} 24 & 79 & 24 \end{bmatrix} \times \begin{bmatrix} 26 \\ 31 \\ 47 \end{bmatrix} = 24 \times 26 + 79 \times 31 + 24 \times 47$$



void destroyMatrix(int** mat);

- Use delete operator to release the dynamic memory space of a given matrix



int printMatrix(int**);

- Print every elements of the matrix