



- Any Questions (Pointers and Arrays)
- Interesting question was asked: “Why accessing array values via pointer arithmetic is important when we can access array values via subscripting?”

Int A[10];

➤ **A[5]**

- Fetch base address of 'A'
- Multiply 5 by data type of 'A', i.e., $5 * 4$
- Add result in base address of A, i.e., base address + 20

Multiplication

Addition

➤ ***(A+5)**

- Fetch pointer 'A'
- Add 5 in pointer A,

Addition

- The process of converting one predefined type into another is called as type conversion
- C++ facilitates the type conversion into the following two forms :
 - ☐ Implicit Type Conversion
 - ☐ Explicit Type Conversion



Implicit Type Conversion

- Conversion performed by the compiler without programmer's intervention whenever differing data types are intermixed in an expression
- The value of the right side (expression side) of the assignment is converted to the type of the left side (target variable)

- Example:

```
int main()
{
    short int x = 1417;
    char ch;
    ch = x;          // where ch is char (1 byte) and x is int (2 bytes)
    return 0;
}
```



Implicit Type Conversion

- **x** was having value 1417 (whose binary equivalent is 0000010110001001)
- **ch** will have lower 8-bits i.e., 10001001 resulting in loss of information.

137





Implicit Type Conversion

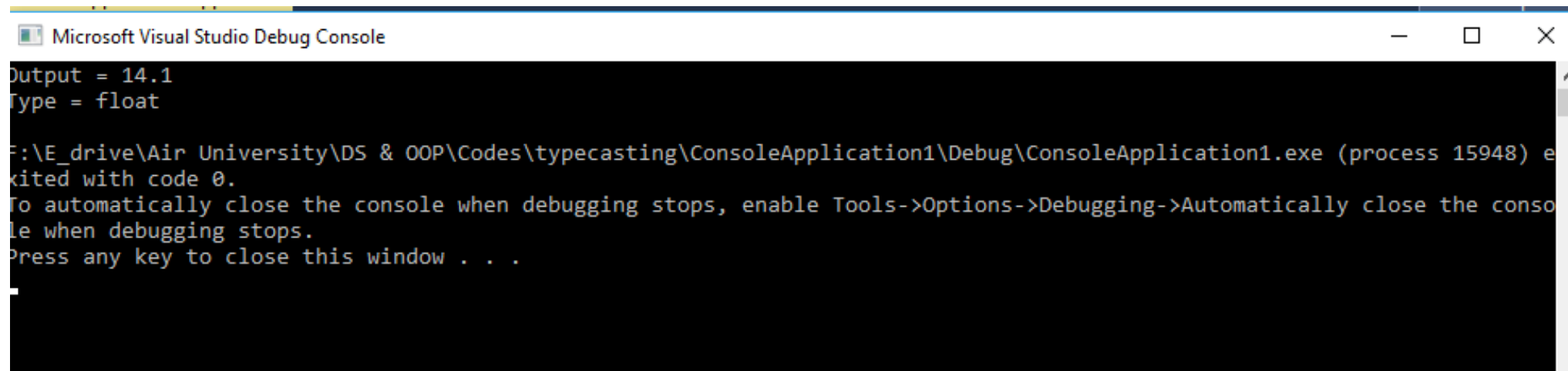
- Another example

```
int main()
{
    int x = 10;

    float y = 4.1;

    cout << "Output = " << x + y << endl;

    cout << "Type = " << typeid(x + y).name() << endl;
}
```



Microsoft Visual Studio Debug Console

```
Output = 14.1
Type = float

F:\E_drive\Air University\DS & OOP\Codes\typecasting\ConsoleApplication1\Debug\ConsoleApplication1.exe (process 15948) e
xited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```



Explicit Type Conversion

- User-defined conversion that forces an expression to be of specific type

```
int main()
{
    int y = 3;
    cout << (float)(y) / 2;
}
```

Output= 1.5

```
int main()
{
    int y = 3;
    cout << (y) / 2;
}
```

Output= 1



- Dynamic memory allocation is necessary because, during compile time, we may not know the exact memory needs to run the program.

new

malloc()

- C++ also does not have automatic garbage collection. Therefore a programmer must manage all dynamic memory used during the program execution

delete[]

free()



Memory Allocation

new / delete[]

```
int main()
{
    int *x;
    x = new int[11];

    for (int i = 0; i <= 10; i++)
        x[i] = 0.1*i;

    delete[] x;
}
```

Return same pointer type

Is a operator

Allocate memory and calls constructor for initialization

malloc() / free() / realloc()

```
int main()
{
    int *x;
    x = (int*) malloc(11 * sizeof(int));

    for (int i = 0; i <= 10; i++)
        x[i] = 0.1*i;

    free(x);
}
```

return void *

stdlib function

Allocate memory and Does not calls constructor

new / delete[]

```
int main()
{
    int rowCount = 10;
    int colCount = 10;

    int** a = new int*[rowCount];
    for (int i = 0; i < rowCount; ++i)
        a[i] = new int[colCount];

    for (int i = 0; i < rowCount; ++i)
        delete[] a[i];
    delete[] a;
}
```

malloc() / free()

YOUR TURN

Thanks a lot



If you are taking a Nap, **wake up**.....Lecture Over