# CSC 215

Math and Computer Science

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

# What size?

- You need to store the numbers from a file into an array.
- How big would you make it????
- Would love to do:

```
int size;
cin >> size;
int array[size];                    // but this is illegal
```

# Guesses?

- 100
  - I have 1,000,000 numbers in the file
- 1000000
  - I only need to store 10 numbers

- The solutions is dynamic memory.

# New Operator

- You can allocate more than one item with the new operator
- Requires an integer to specify how large to create the array
- Allows us to create larger arrays than those taught in csc150.
- The array is placed in heap memory and you are given an address to access it.

Syntax:

```
<dataptr> = new  (nothrow)  <datatype> [integerSize];
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

# Examples

```
int size;
int *iptr = nullptr;
cin >> size;
iptr = new (nothrow) int [size];

int size;
float *fptr = nullptr;
cin >> size;
fptr = new (nothrow) float [size];
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

# There is no guarantees in life

- Just because you want an array does not mean you will be given the storage for that array.
- How much Ram do you have in your computer?
  - 512MB, 1GB, 2GB, 4GB, 8GB, 16GB.

- If the system doesn't have the memory, it will return a nullptr address to you.

# Memory Allocation Check

- Make sure you get the memory.

```
int *ptr;
int size;

cin >> size;
int *ptr;
ptr = new (nothrow) int [size];
if( ptr == nullptr )
{
    cout << "memory allocation error" << endl;
    exit(1);
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

# Use Like Static Arrays

```
for( i=0; i<size; i++)
    ptr[i] = rand();

sum = 0;
for( i=0; i<size; i++)
    sum += ptr[i];
```

# Pass to function expecting static array

```
fillArray( ptr, size );


void fillArray( int ptr[], int size)
{
    int i;
    for( i=0; i<size; i++)
        ptr[i] = rand();
}
```

# Pass to function expecting dynamic array

```
theSUm = sumArray( ptr, size );

int sumArray( int *ptr, int size)
{
    int sum = 0;
    int i;
    for( i=0; i<size; i++ )
        sum += ptr[i];
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

# Freeing the Array

- When you are done, you must free the memory up.

```
delete [] ptr;

// incorrect, frees the 1st data member only
// not the entire array.
delete ptr;
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

# Returning an Array

- You can return a pointer to your array

```
int *myArray = nullptr;
myArray = alloc1D( 1000 );
if( myArray == nullptr )
{   cout << "Cleanup and exit. Memory allocation Failed" << endl;
    exit(1);
}
int *alloc1D(int size)
{   int *ptr = nullptr;
    ptr = new (nothrow) int [size];
    return ptr;
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY