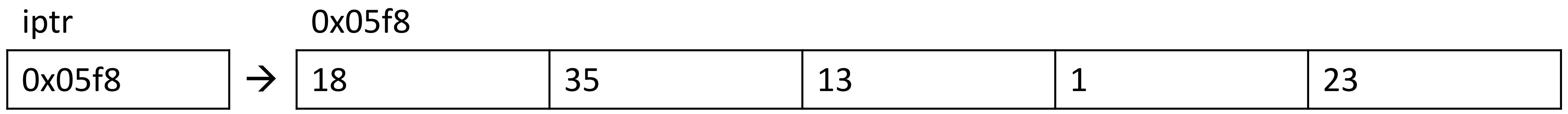
CSC215

Math and Computer Science

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

A Legacy of Excellence

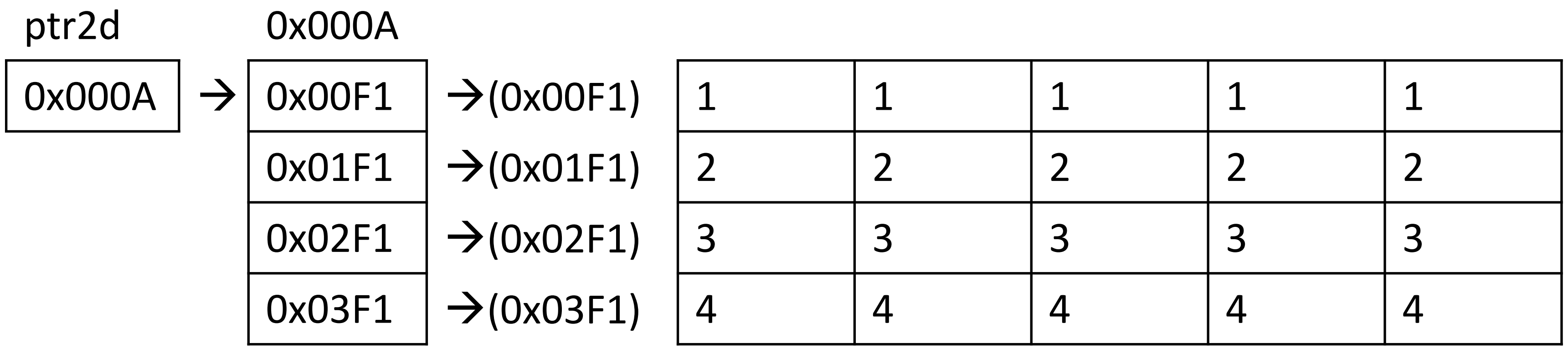
1d Concept



- Have a pointer named iptr (int *).
- Iptr contains the address of the array
- Iptr[i] takes the offset and times the data size and adds it to the address

2D Concept

- Have a pointer to an array of pointers
- Each element in the array of pointers is a pointer to a 1d array



Creating a Pointer to a Pointer

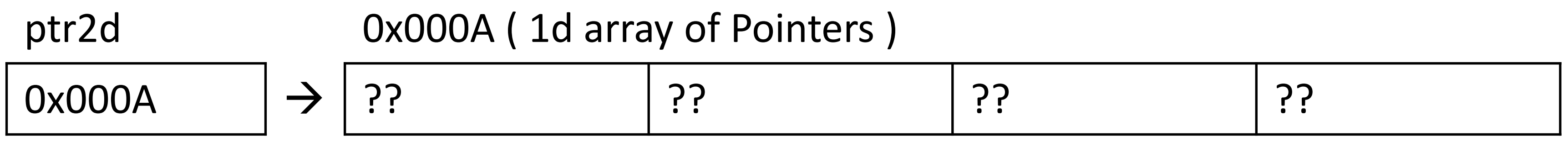
- You need a pointer to a pointer
  - int **iptr;
  - double **dptr;
  - float **fptr;
- The two stars mean it must be dereferenced twice to get to the storage of the data type
  - The first dereference gets you to a single pointer.
  - The second dereference gets you to a data item

Creating the 2d Array

- Create a 2d array of doubles that has 4 rows and 5 columns.

1. Dynamically allocate a 1d array of pointers

```
double **ptr2d = nullptr;
ptr2d = new (nothrow) double * [4];
```



Creating the 2d Array

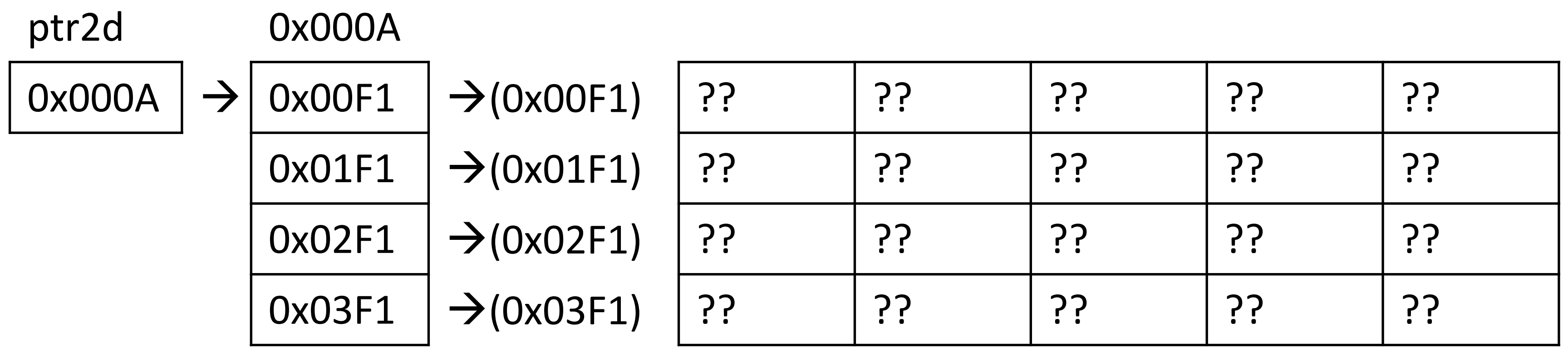
2. Next, go through every single pointer and allocate the storage for the row

```
for( i=0; i<row; i++)
    ptr2d[i] = new (nothrow) double [5];
```

```
This is the first dereference, gets us into
the array of pointers.
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Visually



Now you can use it just like a 2d array

```
for( i=0; i<4; i++)
    for( j=0; j<5; j++)
        ptr2d[i][j] = 1.0;
```

Second dereference

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Not guaranteed Success

```
// could fail here
ptr2d = new (nothrow) double * [4];

for( i = 0; i < 4; i++)
    // could fail here
    ptr2d[i] = new (nothrow) double [5];
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Not guaranteed Success

```
ptr2d = new (nothrow) double * [4];
if( ptr2d == nullptr )
{   cout << "Memory Allocation Error" << endl;
    exit(1);
}
for( i = 0; i < 4; i++)
    // could fail here
    ptr2d[i] = new (nothrow) double [5];
```

Not guaranteed Success

```
ptr2d = new (nothrow) double * [4];
if( ptr2d == nullptr )
{   cout << "Memory Allocation Error" << endl;
    exit(1);
}
for( i = 0; i < 4; i++)
{    ptr2d[i] = new (nothrow) double [5];
     if( ptr2d[i] == nullptr )
     {    for( j=0; j<i; j++)
              delete[] ptr2d[j];
          delete [] ptr2d;
          cout << "Memory Allocation Error" << endl;
          exit(1);
     }
}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Not guaranteed Success

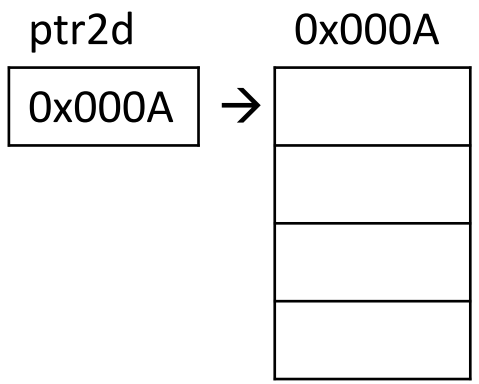
```
ptr2d = new (nothrow) double * [4];
if( ptr2d == nullptr )
{   cout << "Memory Allocation Error" << endl;
    exit(1);
}
for( i = 0; i < 4; i++)
{    ptr2d[i] = new double [5];
     if( ptr2d[i] == nullptr )
     {    for( j=0; j<i; j++)
              delete[] ptr2d[j];
          delete[] ptr2d;
          cout << "Memory Allocation Error" << endl;
          exit(1);
     }}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Freeing the Entire array

- Delete every row of storage

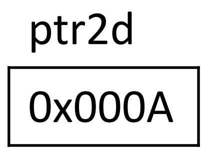
```
for( i=0; i<rows; i++ )
          delete [] ptr2d[i];
```



Freeing the Entire array

- Delete the array of pointers

```
delete [] ptr2d;
```



Freeing the Entire array in a function

```
void free2d( double **&ptr, int rows )
{
    int i;
    if( ptr == nullptr )
        return;

    for( i=0; i<rows; i++ )
        delete [] ptr[i];

    delete [] ptr;
}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Putting the Allocation into a function

```
void alloc2d( double **&ptr,
                  int row, int cols)
{
    ptr = new (nothrow) double * [row];
    if( ptr == nullptr )
    {
        return nullptr;
    }
```

```
    for( i=0; i<row; i++)
    {
        ptr[i] = new (nothrow) double [cols];
        if( ptr[i] == nullptr )
        {
            free2d( ptr, i );
            return;
        }
    }
}
```

Using the functions

```
double **lowtemps = nullptr;
double **hightemps = nullptr;

alloc2d( lowtemps, 12, 31 );
alloc2d( hightemps, 12, 31 );
if( lowtemp == nullptr || hightemp == nullptr)
{      free2d( lowtemp, 12 );
       free2d( hightemp, 12 );
       cout << "Memory allocation Failed" << endl;
       exit (1);
}
```

SOUTH DAKOTA
SCHOOL OF MINES & TECHNOLOGY

or Use this Function

```
double **alloc2d( int row, int cols)
{
    double **ptr = nullptr;
    ptr = new (nothrow) double * [row];
    if( ptr == nullptr )
    {
        return nullptr;
    }
    for( i=0; i<row; i++)
    {
        ptr[i] = new double [cols];
        if( ptr[i] == nullptr )
        {
            free2d( ptr, i );
            return nullptr;
        }
    }
    return ptr;
}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Using the functions

```
double **lowtemps = nullptr;
double **hightemps = nullptr;

lowtemps = alloc2d( 12, 31 );
hightemps = alloc2d( 12, 31 );

if( lowtemp == nullptr || hightemp == nullptr)
{     free2d( lowtemp, 12 );
      free2d( hightemp, 12 );
      cout << "Memory allocation Failed" << endl;
      exit (1);
}
```

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Passing 2d arrays (dynamic) to functions

- Dynamic 2d arrays are more flexible than static 2d arrays.
- It is good practice to pass the row and column with the pointer
- Examples: printMatrix(iptr, 10, 20);
  printMatrix(iptr2, 100, 1000);

```
void printMatrix( int **ptr, int r, int c)
{   int i,j;
    for( i=0; i<r; i++)
    {   for( j=0; j<c; j++)
            cout << ptr[i][j] << " ";
        cout << endl;
}   }
```

SOUTH DAKOTA
SCHOOL OF MINES & TECHNOLOGY