CSC215

Math and Computer Science

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Pointers

- What are they
  - Contain a memory address
  - Value refers directly to another value stored somewhere else in memory
- Syntax:

  datatype * identifiername;

Examples:

```
int * intptr;
double * dptr;
char * str;
```

Addresses

- Every variable has a memory address associated with it for storage
- This is how we pass a variable by reference
- To get the address of a variable, use the & (address of, reference operator)

Example

```
int *ptr = nullptr;          // good practice to initialize ptrs to null
int x = 25;

ptr = &x;                    // the address of x is now stored in ptr
```

Dereferencing the Pointer

- To use the memory address, you must dereference the pointer.
- In other words, grab the address stored in the pointer, go to that address in memory and access its contents.

Example

```
int *ptr =nullptr;
int x = 25;                    //top
ptr = &x;
cout << *ptr << endl;          // middle
*ptr = *ptr / 5;               // last
cout << *ptr << endl;
```

Id Name	Address	Value
ptr	0x000a	nullptr
	:	
x	0x00F0	25

Id Name	Address	Value
ptr	0x000a	0x00F0
	:	
x	0x00F0	25

Id Name	Address	Value
ptr	0x000a	0x00F0
	:	
x	0x00F0	5

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Example

```
double *dptr =nullptr;
double val = 35.0;          //top
dptr = & val;
cout << *dptr << endl;      // middle
*dptr = *dptr / 5.0;        // last
cout << *dptr << endl;
```

Id Name	Address	Value
dptr	0x000a	nullptr
	:	
val	0x0EF0	35.0

Id Name	Address	Value
dptr	0x000a	0x0EF0
	:	
val	0x0EF0	35.0

Id Name	Address	Value
dptr	0x000a	0x0EF0
	:	
val	0x0EF0	7.0

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Pointers vs Variables

float v = 34.4;

float *fptr = &v;

- 2 things in common
  - Both variables have memory locations
    - cout << &v;
    - cout << &fptr;
  - Access the contents using just the name
    - cout << v
    - cout << fptr;
- Pointers have dereferencing

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Passing Pointers to functions

- Pointers are pass by value.
- What they point to is pass by reference.
  - When dereferenced, you change a variable in memory.
  - Be careful about returning a pointer to an automatic variable. The variable would be destroyed when the function exits.

Pass by Value

```
int *ptr = nullptr;
int x = 25;
cout << "main: " << ptr << endl;
value( ptr, x );
cout << "main: " << ptr << endl;
// both main: are null (0x00000000)
void value( int *iptr, int &num )
{
    iptr = &num;
    cout << "function: " << iptr <<endl;
}
```

Value	Main	address	Memory
	ptr	0x00FA	0x0000
num	x	0x01F0	25
Iptr		0x0200	0x01f0

SOUTH DAKOTA
SCHOOL OF MINES
& TECHNOLOGY

Pass by Reference

```
int *ptr = nullptr;
int x = 25;
cout << "main: " << ptr << endl;
reference( ptr, x );
cout << "main: " << ptr << endl;
void reference( int *&iptr, int &num)
{
    iptr = &num;
}
main: 0x0000
main: 0x01F0
```

Main	address	Memory
ptr	0x00FA	0x0000
x	0x01F0	25

Value	Main	address	Memory
iptr	ptr	0x00FA	0x01F0
num	x	0x01F0	25

Main	address	Memory
ptr	0x00FA	0x01F0
x	0x01F0	25

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Pass by Value (what it points to)

```
int x = 25;
int *ptr = &x;
cout << "main: " << x << endl;
valueref( ptr );
cout << "main: " << x << endl;
void valueref( int *iptr)
{
        *iptr = *iptr * 20;
}
```

main: 25

main: 500

Value	Main	address	Memory
	ptr	0x00FA	0x01F0
	x	0x01F0	25
iptr		0x0200	0x01F0

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

New Operator

- You can create a new variable on the fly.
- The new operator returns the address of a new variable.
- Store this address in a pointer
- Syntax:
  - new (nothrow) <datatype>; // or new <datatype>

```
int *ptr = nullptr;
float *fptr = nullptr;
ptr = new (nothrow) int;
fptr = new (nothrow) float;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Delete Operator

- Your responsibility as a programmer to free this data up when you are done.
- To free the memory, you use the delete operator.
- If you do not free the memory up, it is known as a memory leak.
- Syntax:

  delete pointer;

```
delete ptr;
delete fptr;
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Returning a pointer

- You can return an address from a function.

```
int *getNewInteger()
{
    int *p = new (nothrow) int;
    return p;
}
float *getNewFloat()
{
    float *f = new (nothrow) float;
    return p;
}
```

SOUTH DAKOTA SCHOOL OF MINES & TECHNOLOGY

Returning a pointer

- *Never* return an address to a local variable

```
int *badAddress()
{
    int value = 20;
    return &value;
}
// value gets destroyed when the
// function is exited
```