

Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

# Pointers

## Comp Sci 1575 Data Structures



Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

“Writing in C or C++ is like running a chain saw with all the safety guards removed,” Bob Gray.

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence



Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

## Data Structures?

### Definitions

### Addresses

#### Memory

### Pointers

#### Declaration

#### Initialization

#### Dereference

#### Dereference assignment

#### Uses

#### Issues?

#### Pointers to pointers

#### Careful cancellation

### Pointers and arrays

#### What is an array?

#### Array and pointer indexing

### Pointer arithmetic

#### Type sizing

#### Operator precedence

- We are finishing C++ first, and won't start the actual Data Structures material for a while.
- Reserve judgment about how fun you consider the material to be until then!

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 **Definitions**
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

Data  
Structures?

**Definitions**

Addresses

Memory

**Pointers**

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

**Pointers and  
arrays**

What is an array?

Array and pointer  
indexing

**Pointer  
arithmetic**

Type sizing

Operator  
precedence

- A pointer is a variable whose value is the address of another variable.
- What is an address?
- How do you get the memory address of a variable?

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 **Addresses**
  - Memory
- 4 **Pointers**
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 **Pointers and arrays**
  - What is an array?
  - Array and pointer indexing
- 6 **Pointer arithmetic**
  - Type sizing
  - Operator precedence



Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

How do you get the memory address of an int for example?

```
int genePos = 435;
cout << &genePos << endl; // 0x7ffcb158c144
```

- & is the “address of” operator
- What is that weird number?
- How is memory structured?
- It is different than, but similar to the reference type specified with & (more to come later)

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

1 Data Structures?

2 Definitions

3 Addresses

Memory

4 Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

5 Pointers and arrays

What is an array?

Array and pointer indexing

6 Pointer arithmetic

Type sizing

Operator precedence

Data  
 Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

 Dereference  
 assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

 Pointers and  
 arrays

What is an array?

 Array and pointer  
 indexing

 Pointer  
 arithmetic

Type sizing

 Operator  
 precedence

```
int genePos = 435;
cout << &genePos << endl; // 0x7ffcb158c144
```

| Name of variable | Storage address | Value |
|------------------|-----------------|-------|
|                  | 0x7ffcb158c140  |       |
| genePos          | 0x7ffcb158c144  | 435   |
|                  | 0x7ffcb158c148  |       |
|                  | 0x7ffcb158c14c  |       |
|                  | 0x7ffcb158c150  |       |
|                  | 0x7ffcb158c154  |       |

- Variable name is an alias for address itself, which is accessible via the & operator

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

```
int genePos = 435;
int *p1 = &genePos;
cout << p1 << endl; // 0x7ffcb158c144
```

| Name of variable | Storage address | Value          |
|------------------|-----------------|----------------|
|                  | 0x7ffcb158c140  |                |
| genePos          | 0x7ffcb158c144  | 435            |
|                  | 0x7ffcb158c148  |                |
|                  | 0x7ffcb158c14c  |                |
| p1               | 0x7ffcb158c150  | 0x7ffcb158c144 |
|                  | 0x7ffcb158c154  |                |

- p1 is a pointer – a variable whose value is the address of another variable.

Data  
Structures?

Definitions

Addresses

Memory

**Pointers**

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

**Pointers and  
arrays**

What is an array?

Array and pointer  
indexing

**Pointer  
arithmetic**

Type sizing

Operator  
precedence

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 **Pointers**
  - Declaration
  - Initialization
  - Dereference
    - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

# Pointer declaration (type specific)

- Like other variables or constants, pointers must be declared
- General pointer variable declaration is:  
 $type * pointerName = \&varName$   
 where type is the pointer's base type
- Pointers have a type (of the thing they address) restriction (e.g., type is "pointer to an int" or "pointer to a double")
- Can cast between pointer types, e.g., static cast, but should not generally to non-pointer types. This won't usually be needed.

Declaring types of pointers:

```
int *numberObject; // pointer to an int
char *characterObject; // pointer to a char
double *decimalObject; // pointer to a double
```



Data  
Structures?

Definitions

Addresses

Memory

Pointers

**Declaration**

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

```
int *p1, *p2; // both p1 and p2 are pointers
int *p1, p2; // p2 is not a pointer!
```

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

**Initialization**

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 **Pointers**
  - Declaration
  - Initialization**
  - Dereference
    - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

```
int genePos; // what is the value of genePos?
int *p1 = &genePos;
```

```
int genePos;
int *p1; // what does p1 point to?
p1 = &genePos;
```

```
int genePos;
int *p1 = &genePos;
int *p2 = p1;
```

```
// To specify no target
int genePos;
int *p1 = nullptr; //or =NULL, or =0 (old C++)
// Use nullptr for this class!
```

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

**Dereference**

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

## 1 Data Structures?

## 2 Definitions

## 3 Addresses

Memory

## 4 Pointers

Declaration

Initialization

**Dereference**

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

## 5 Pointers and arrays

What is an array?

Array and pointer indexing

## 6 Pointer arithmetic

Type sizing

Operator precedence

# Contents of object being pointed to

```

int genePos = 435;
int *p1 = &genePos;
cout << *p1 << endl; // outputs: 435
int x = *p1;
cout << x << endl; // outputs: 435
    
```

| Name of variable | Storage address | Value          |
|------------------|-----------------|----------------|
|                  | 0x7ffcb158c140  |                |
| genePos          | 0x7ffcb158c144  | 435            |
|                  | 0x7ffcb158c148  |                |
|                  | 0x7ffcb158c14c  |                |
| p1               | 0x7ffcb158c150  | 0x7ffcb158c144 |
|                  | 0x7ffcb158c154  |                |

- Contents of operator also known as dereference operator, \*
- This is not the same as the \* used during initialization; the \* on lines 2 and 3 are different

# Assignment via dereferenced pointer

- Data
- Structures?
- Definitions
- Addresses
- Memory
- Pointers
- Declaration
- Initialization
- Dereference
- Dereference assignment**
- Uses
- Issues?
- Pointers to pointers
- Careful cancellation
- Pointers and arrays
- What is an array?
- Array and pointer indexing
- Pointer arithmetic
- Type sizing
- Operator precedence

```

int genePos = 435;
int *p1 = &genePos;
cout << *p1 << endl; // outputs: 435
*p1 = 248;
cout << *p1 << endl; // outputs: 248
    
```

| Name of variable | Storage address | Value              |
|------------------|-----------------|--------------------|
|                  | 0x7ffcb158c140  |                    |
| genePos          | 0x7ffcb158c144  | 435 changed to 248 |
|                  | 0x7ffcb158c148  |                    |
|                  | 0x7ffcb158c14c  |                    |
| p1               | 0x7ffcb158c150  | 0x7ffcb158c144     |
|                  | 0x7ffcb158c154  |                    |

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

Data  
 Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
 assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
 arrays

What is an array?

Array and pointer  
 indexing

Pointer  
 arithmetic

Type sizing

Operator  
 precedence

- Used for new memory during execution, e.g., dynamic memory
- Which is useful for what?  
 User-defined quantities or sized of objects
- Can refer/pass large data structures without copying, for efficiency
- Can specify relationships among data, e.g., linked lists, trees, graphs, coming up



Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

## 1 Data Structures?

## 2 Definitions

## 3 Addresses

Memory

## 4 Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

**Issues?**

Pointers to pointers

Careful cancellation

## 5 Pointers and arrays

What is an array?

Array and pointer indexing

## 6 Pointer arithmetic

Type sizing

Operator precedence

Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

With great power comes great responsibility! You can break more fundamental things with pointers... Get ready for segfaults.

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

**Pointers to pointers**

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 **Pointers**
  - Declaration
  - Initialization
  - Dereference
    - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers**
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 **Pointer arithmetic**
  - Type sizing
  - Operator precedence

```

int genePos = 435;
int *p1 = &genePos;
int **metaP = &p1;
cout << metaP << endl; // ??
cout << *metaP << endl; // ??
cout << **metaP << endl; // ??
    
```

| Name of variable | Storage address | Value          |
|------------------|-----------------|----------------|
|                  | 0x7ffcb158c140  |                |
| genePos          | 0x7ffcb158c144  | 435            |
|                  | 0x7ffcb158c148  |                |
|                  | 0x7ffcb158c14c  |                |
| p1               | 0x7ffcb158c150  | 0x7ffcb158c144 |
| metaP            | 0x7ffcb158c154  | 0x7ffcb158c150 |

Remember, \*\* for declaring and dereferencing are different, as are the two \*

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

# What about these statements?

 Data  
 Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

 Dereference  
 assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

 Pointers and  
 arrays

What is an array?

 Array and pointer  
 indexing

 Pointer  
 arithmetic

Type sizing

 Operator  
 precedence

```
cout << &*p1 << endl; // ??
```

```
cout << *&p1 << endl; // ??
```

```
cout << &*&p1 << endl; // ??
```

```
cout << *&*&p1 << endl; // ??
```

| Name of variable | Storage address | Value          |
|------------------|-----------------|----------------|
|                  | 0x7ffcb158c140  |                |
| genePos          | 0x7ffcb158c144  | 435            |
|                  | 0x7ffcb158c148  |                |
|                  | 0x7ffcb158c14c  |                |
| p1               | 0x7ffcb158c150  | 0x7ffcb158c144 |
| metaP            | 0x7ffcb158c154  | 0x7ffcb158c150 |

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

**What is an array?**

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?**
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence



- Data Structures?
- Definitions
- Addresses
  - Memory
- Pointers
  - Declaration
  - Initialization
  - Dereference
    - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- Pointers and arrays
  - What is an array?**
  - Array and pointer indexing
- Pointer arithmetic
  - Type sizing
  - Operator precedence

Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

**What is an array?**

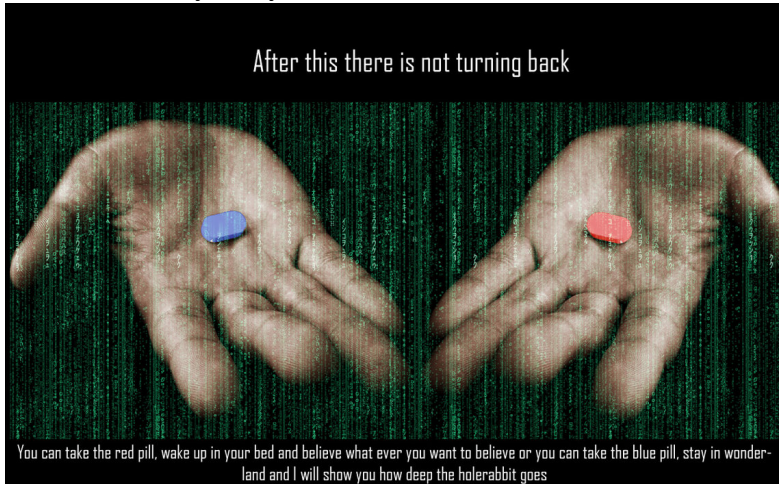
Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

## What is an array really?



Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

**What is an array?**

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

What is an array really?

```

int a[6] = {1,7,3,4,2,8};
cout << a[2] << endl; // outputs: 3
cout << a << endl; // ??
cout << *a << endl; // ??
  
```

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- Arrays are like pointers, but const, addressing the first element of the array
- Below, mypointer can be assigned a different address, but myarray can't.

```
int myarray[20];
cout << myarray << endl; // 0x7ffcb158c140
```

```
int *mypointer;

// Valid , why no & operator before myarray?
// Recall passing arrays by reference?
mypointer = myarray;
cout << mypointer << endl; // 0x7ffcb158c140

// Invalid , why?
myarray = mypointer;
```

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

- The following have the same result:

```
int a[6] = {1, 7, 3, 4, 2, 8};
```

```
a[5] = 0;           // a [offset of 5] = 0
cout << a[5] << endl; // outputs: 0
cout << *(a + 5) << endl; // outputs: 0
```

```
*(a+5) = 1;        // a [offset of 5] = 1
cout << a[5] << endl; // outputs: 1
cout << *(a + 5) << endl; // outputs: 1
```

Why does adding 5 to array a work?

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence

- 1 Data Structures?
- 2 Definitions
- 3 Addresses
  - Memory
- 4 Pointers
  - Declaration
  - Initialization
  - Dereference
  - Dereference assignment
  - Uses
  - Issues?
  - Pointers to pointers
  - Careful cancellation
- 5 Pointers and arrays
  - What is an array?
  - Array and pointer indexing
- 6 Pointer arithmetic
  - Type sizing
  - Operator precedence

```

int a[6] = {1,7,3,4,2,8};
int *pa = a;
cout << pa + 2 << endl; // 0x7ffcb158c148
cout << *(pa + 2) << endl; // 3
cout << pa++ << endl; // 0x7ffcb158c144
cout << *pa << endl; // 7
    
```

| Name of variable | Storage address | Value          |
|------------------|-----------------|----------------|
| a[0] or *a       | 0x7ffcb158c140  | 1              |
| a[1] or *(a+1)   | 0x7ffcb158c144  | 7              |
| a[2] or *(a+2)   | 0x7ffcb158c148  | 3              |
| a[3] or *(a+3)   | 0x7ffcb158c14c  | 4              |
| a[4] or *(a+4)   | 0x7ffcb158c150  | 2              |
| a[5] or *(a+5)   | 0x7ffcb158c154  | 8              |
| a                | 0x...           | 0x7ffcb158c140 |
| pa               | 0x...           | 0x7ffcb158c140 |

- Why increments of 4?



Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

```
int *pa = a;
cout << pa + 2 << endl; // 0x7ffcb158c148
cout << a + 2 << endl; // 0x7ffcb158c148
```

```
cout << *(pa+2) << endl; // 3
cout << *(a+2) << endl; // 3
```

```
cout << pa[2] << endl; // 3
cout << a[2] << endl; // 3
```

```
cout << pa++ << endl; // 0x7ffcb158c144
//cout << a++ << endl; // not valid, array const
```

```
cout << *pa << endl; // 7
cout << *a << endl; // 1
```

Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

Operator  
precedence

# Each type is a different size

Data Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and arrays

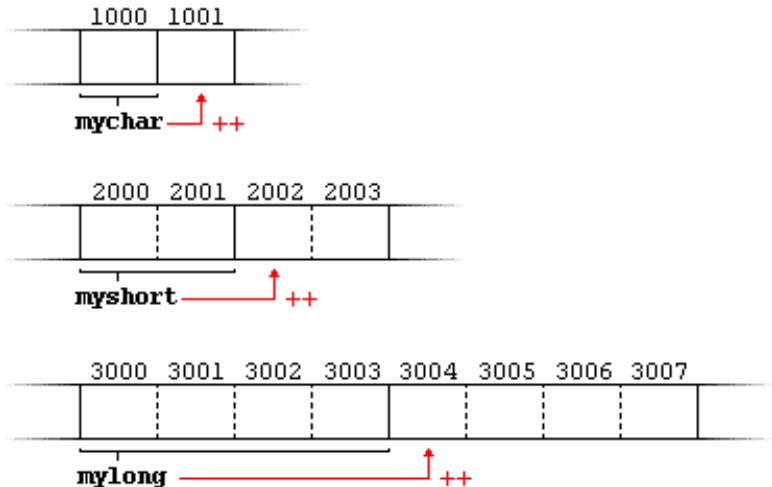
What is an array?

Array and pointer indexing

Pointer arithmetic

Type sizing

Operator precedence



Use `sizeof(p)` without the `*` operator to determine the memory utilized on your system for types like `int`, which are different per system.

Postfix operators (`--`, `++`), have higher precedence than prefix operators (dereference `*`).

```

int genePosArray[3] = {435,123,987};
int *p = genePosArray;
cout << p << endl; // 0x7ffe35b36ee0
cout << *(p++) << p << endl; // 435 0x7ffe35b36ee4
p = genePosArray;
cout << *p++ << p << endl; // 435 0x7ffe35b36ee4
p = genePosArray;
cout << *(++p) << p << endl; // 123 0x7ffe35b36ee4
p = genePosArray;
cout << *++p << p << endl; // 123 0x7ffe35b36ee4
p = genePosArray;
cout << ++(*p) << p << endl; // 436 0x7ffe35b36ee0
p = genePosArray;
cout << ++*p << p << endl; // 437 0x7ffe35b36ee0
p = genePosArray;
cout << (*p)++ << p << endl; // 437 0x7ffe35b36ee0
  
```

Data  
Structures?

Definitions

Addresses

Memory

Pointers

Declaration

Initialization

Dereference

Dereference  
assignment

Uses

Issues?

Pointers to pointers

Careful cancellation

Pointers and  
arrays

What is an array?

Array and pointer  
indexing

Pointer  
arithmetic

Type sizing

**Operator  
precedence**

Dynamic memory (heap, stack, garbage collection, dangling pointers), pointers to classes and structs, const pointers, arrays of pointers, void pointers, pointers to functions, returning pointers from functions