Data

Definition

Address

Pointer

Declaration

Initializatio

Dereference

assignmer

Uses

Pointers to pointers

Pointers and arrays

What is an array? Array and pointer indexing

Pointer arithmetic

Operator

Pointers

Comp Sci 1575 Data Structures



Data

.

Definition

Addres

Doclarati

Initializat

Dereferen

Derefere

Uses

Issues?

Pointers to pointer

Careful cancellation

Pointers and arrays

What is an array?

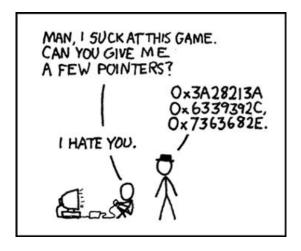
Array and pointer

Pointer arithmetic

Operator

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





Data Structures?

Definitions

Address

Pointe

Declaration

Initializatio

Doroforon

Derefe

assign

Uses

Pointers to point

Careful cancellati

Pointers and arrays

Array and pointer indexing

arithmetic

Operator

Data Structures?

2 Definitions

3 Addresses

Memory

4 Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointer

Careful cancellation

6 Pointers and arrays

What is an array?

6 Pointer arithmetic

Type sizing

Operator precedence

Data Structures yet?

Data Structures?

Definition

Address

Memory

Pointe

Declaration

Dereference

Dereferer assignme

Uses

Pointers to pointers
Careful cancellation

Pointers and arrays

What is an array?
Array and pointer indexing

Pointer arithmetic

- 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

Data Structures?

2 Definitions

3 Addresses
Memory

4 Pointers

Declaration Initialization

Dereference

Dereference assignment

Issues?

Pointers to pointers

Careful cancellation

6 Pointers and arrays

What is an array?

Array and pointer indexing

6 Pointer arithmetic

Type sizing

Operator precedence

Memory

Pointe

Declaration

Initializati

Dereferen

assig

Uses

Careful cancell

arrays
What is an array

Array and pointer indexing

arithmeti Type sizing

Operator precedent



Definition of a pointer

Data Structures

Definitions

Memony

Pointe

Declarat

Dereferen

Dereterei assignme

Uses

Pointers to pointer

Pointers and arrays

What is an array?
Array and pointer indexing

Pointer arithmetic

- 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?

Addresses

Addresses

```
Data
Structures?
```

Addresses

Pointer

Declaration Initializatio

Dereference

Uses

Pointers to pointers

Pointers and arrays
What is an array?
Array and pointer

Pointer arithmetic _{Type sizing} How do you get the memory address of an int for example?

```
int genePos = 435; cout << &genePos << endI; // 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)

Addresses

Memory

Memory

Structures

Addresse

Declaration
Initialization
Dereference
Dereference
assignment

Issues?
Pointers to pointers
Careful cancellation

Pointers and arrays
What is an array?
Array and pointer

Pointer
arithmetic
Type sizing
Operator
precedence

int genePos = 435; cout << &genePos << endI; // 0x7ffcb158c144

Name of variable	Storage address	Value
	0×7ffcb158c140	
genePos	0x7ffcb158c144	435
	0x7ffcb158c148	
	0x7ffcb158c14c	
	0x7ffcb158c150	
	0x7ffcb158c154	

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

Data Structures

Data Structures

2 Definitions

3 Addresses Memory

4 Pointers

Declaration Initialization

Dereference

Dereference assignm

Uses

Issues

Pointers to pointers

6 Pointers and arrays

What is an array? Array and pointer indexing

6 Pointer arithmetic

Type sizing
Operator precedence

Delate

Declaration

Initializati

Derefer

Uses Issue:

Pointers to point Careful cancellat

arrays

Array and pointer indexing

arithmeti Type sizing

```
Data
Structures
```

Addresses

Pointers Declaration

Careful cancellati Pointers and arrays

```
Pointer
arithmetic
Type sizing
```

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

	6	
Name of variable	Storage address	Value
	0×7ffcb158c140	
genePos	0×7ffcb158c144	435
	0×7ffcb158c148	
	0x7ffcb158c14c	
p1	0x7ffcb158c150	0x7ffcb158c144
	0x7ffcb158c154	

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

Definition

Addres

Memory

Pointers

Declaration

Initializatio

Danisana

Dereferenc

assignm

Issues?

Pointers to pointers

Pointers and

What is an array

Array and pointe indexing

arithmeti

Pointers

Declaration

Pointer declaration (type specific)

Data Structures?

Addresses Memory

Declaration

Uses Issues? Pointers to pointers

Pointers and arrays
What is an array?
Array and pointer indexing

Pointer arithmetic Type sizing

- 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
```

Pointer declaration

Data

Definitio

Addresse

. . .

Declaration

Declaration

Initializati

Derefere

assignme

Uses

Pointers to pointe

Pointers and

What is an array?
Array and pointer

Pointer arithmetic

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

Data

O Definitions

2 Addresses

Addresses Memory

4 Pointers

Declaration

Initialization

Dereference

Dovefore

Dereference assignment

Uses

Issues

Pointers to pointer

Careful cancellation

5 Pointers and arrays

What is an array?

Array and pointer indexing

6 Pointer arithmetic

Type sizing

Definiti

Memory

Pointer

Initializati

Derefere

Uses Issues

Careful cancellat

arrays
What is an arra

Array and pointer indexing

arithmeti Type sizing

```
Computer Scient
```

Pointer initialization variations

```
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

Definitions

Address

3 Addresses

Memory

4 Pointers

Declaration Initialization

Dereference

Dereference assignmen

Uses

Issues

Pointers to pointers

Careful cancellation

5 Pointers and arrays

What is an array?

6 Pointer arithmetic

Гуре sizing Operator precedence

_ .

Pointer

Initializati

Derefer

assigi Uses

Pointe

Pointers a arrays

Array and pointer indexing

arithmeti Type sizing

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

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

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

```
Computer Scier
```

Assignment via dereferenced pointer

```
Data
Structures?
Definitions
```

Addresses Memory

Declaration Initialization

Dereference

Careful cancellation
Pointers and

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</pre>
```

Name of variable	Storage address	Value
	0×7ffcb158c140	
genePos	0×7ffcb158c144	435 changed to 248
	0x7ffcb158c148	
	0x7ffcb158c14c	
p1	0×7ffcb158c150	0x7ffcb158c144
	0×7ffcb158c154	

Data

Data Struct

Address

3 Addresses

Memory

4 Pointers

Declaration

Initialization

Dereference

Dereference assignment

Uses

Issues?

Pointers to pointers

5 Pointers and arrays

What is an array?
Array and pointer indexing

6 Pointer arithmetic

Type sizing Operator precedence

Pointe

Derefi assign Uses

Issues?

Careful cancellat

arrays
What is an array?
Array and pointer

Pointer

Type sizing

Address

Pointe

Initialization Dereference

Dereference assignment

Issues?

Pointers to pointers Careful cancellation

arrays
What is an array?
Array and pointer

Pointer arithmetic

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

Pointers

Issues?



Pointers are dangerous

Data

.

Address

Memory

Pointei

Initializati

Dereference

assignme

Issues?

Pointers to pointers Careful cancellation

Pointers and arrays

What is an array? Array and pointer indexing

Pointer arithmetic

Operator

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

Data Structures Data Structures?

2 Definitions

3 Addresses Memory

4 Pointers

Declaration Initialization

Dereference

Dereference assignment

Uses Issues?

Pointers to pointers

Careful cancellation

What is an array?

6 Pointer arithmetic

Type sizing

Type sizing Operator precedence

Address

Memory

Pointer

Initializatio

Dereferend

Uses

Pointers to

Pointers an arrays

Array and pointer indexing

arithmeti Type sizing

```
Data
tructures?
Definitions
addresses
Memory
Tointers
Declaration
initialization
Dereference
Dereference
Susses?
Designers to pointers
```

```
Pointers to pointers
Careful cancellation
```

```
arrays
What is an array?
Array and pointer indexing
Pointer
arithmetic
```

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

Name of variable	Storage address	Value
	0×7ffcb158c140	
genePos	0×7ffcb158c144	435
	0×7ffcb158c148	
	0x7ffcb158c14c	
p1	0x7ffcb158c150	0x7ffcb158c144
metaP	0x7ffcb158c154	0x7ffcb158c150

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

Data Structures Data Structures?

2 Definitions

3 Addresses Memory

4 Pointers

Declaration Initialization

Dereference

Dereference assignment

Uses

Issues!

Pointers to pointer

Careful cancellation

5 Pointers and arrays

What is an array?

6 Pointer arithmetic

Гуре sizing Operator precedence

Memory

Declaration

Initializati

Deref assign

Uses

Pointers to point

Careful cancellat

Pointers and arrays

Array and pointer indexing

arithmeti Type sizing

What about these statements?

```
Data
Structures?
Definitions
```

Addresses

Declaratio Initializati Dereferen

> Uses Issues? Pointers to pointe

Careful cancellation
Pointers and
arrays

Array and pointer indexing

Pointer

```
Type sizing
Operator
```

```
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

Structures?

Address

Pointers

Declaratio

Dereference

assign

Issues

Pointers to point Careful cancellati

Pointers and arrays

What is an array? Array and pointer indexing

Pointer arithmetic _{Type sizing}

Operator

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

5 Pointers and arrays What is an array?

What is an array?

Definition

Address

Memory

Fointers

1 20 10 10

Initializatio

Dereterence

assignment

Uses

Pointers to pointer

Pointers and

arrays
What is an array?

Array and pointer

arithmeti



Pointers and arrays

Data

Definition

Address

Pointer

Pointe

Initializati

Derefere

assignm

Uses Issues?

Pointers to pointe

Pointers and arrays
What is an array?

Array and pointer indexing

arithme

Operator

What is an array really?



You can take the red pill, wake up in your bed and believe what ever you want to believe or you can take the blue pill, stay in wonderland and I will show you how deep the holerabbit goes

Data

.....

Memory

Pointer

Initializatio

Dereference

Dereference

assignment

Uses

Pointers to pointe

Pointers and arrays

What is an array?
Array and pointer indexing

Pointer arithmetic

Operator

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; // ??</pre>
```

What is an array?

```
• 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 \ll myarray \ll 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 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 arrav?

Array and pointer indexing

6 Pointer arithmetic

Type sizing

Operator precedence

Memory

Declaratio

Initializati

Deref assign

Uses

Pointers to poin Careful cancella

Pointers an arrays

Array and pointer indexing

arithmetic

Operator preceden

```
Data
Structures
```

D-f:=:±:---

Addresse

Pointer

Declaration Initialization Dereference Dereference

Uses Issues?

Pointers to pointers Careful cancellation

arrays
What is an array?
Array and pointer

Pointer arithmetic Type sizing

Operator precedent • The following have the same result:

```
int a[6] = \{1, 7, 3, 4, 2, 8\};
a[5] = 0; // a[offset of 5] = 0
cout \ll a[5] \ll endl; // outputs: 0
cout \ll *(a + 5) \ll endl; // outputs: 0
*(a+5) = 1; // a [offset of 5] = 1
cout \ll a[5] \ll endl; // outputs: 1
cout \ll *(a + 5) \ll endl; // outputs: 1
Why does adding 5 to array a work?
```

Data Structures

Data Structo

2 Definitions

3 Addresses Memory

4 Pointers

Declaration Initialization

Dereference

Dereference assignment

Uses Issues?

Pointers to pointers

Careful cancellation

6 Pointers and arrays

What is an array? Arrav and pointer indexing

6 Pointer arithmetic

Type sizing Operator precedence

Address Memory

Pointe

Declaration Initializat

Dereferer

assig Uses

Issues

Pointers ar

What is an array? Array and pointer indexing

Pointer arithmetic

```
;?
5
```

Pointer arithmetic

```
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	0×7ffcb158c140	1
a[1] or *(a+1)	0x7ffcb158c144	7
a[2] or *(a+2)	0×7ffcb158c148	3
a[3] or *(a+3)	0x7ffcb158c14c	4
a[4] or *(a+4)	0×7ffcb158c150	2
a[5] or *(a+5)	0x7ffcb158c154	8
a	0x	0x7ffcb158c140
pa	0x	0x7ffcb158c140

Why increments of 4?

Pointer and array arithmetic

```
Data
Structures
```

Definitions

Addresses

Pointer

Declaration Initialization Dereference

assignn Uses

Pointers to pointers
Careful cancellation

Pointers and arrays What is an array? Array and pointer indexing

Pointer arithmetic Type sizing Operator

```
int *pa = a:
cout << pa + 2 << endl; // 0x7ffcb158c148
cout << a + 2 << endl; // 0x7ffcb158c148
cout << *(pa+2) << endl; // 3
cout << *(a+2) << endl; // 3
cout \ll pa[2] \ll endl; // 3
cout << a[2] << endl; // 3
cout << pa++ << endl; // 0x7ffcb158c144
//cout \ll a++ \ll endl; // not valid, array const
cout << *pa << endl; // 7
cout << *a << endl: // 1
```

Data

Church

Definition

Addres

Memory

Pointers

Declaration

Initialization

Dereference

Dereferenc

Uses

Pointers to pointers

Careful cancellation

Pointers and arrays

What is an array?
Array and pointer

Pointer arithmetic

Type sizing

Operator precedence

Each type is a different size

Data Structures?

Addresse

Pointe

Initialization Dereference

assignment
Uses
Issues?

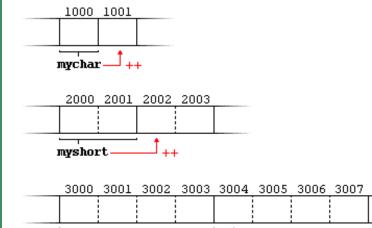
Pointers to pointer Careful cancellation

Pointers and arrays
What is an array?

Pointer arithmetic

mylong

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.

```
Computer Scie
```

Messy pointer arithmetic

```
Data
Structures?
Definitions
Addresses
Memory
Pointers
Declaration
```

Pointer

arithmetic

Operator precedence

```
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 0x7ffe35b36e
```

```
cout << *(p++) << p << endl; // 435 <math>0x7ffe35b36ee4
p = genePosArray;
cout << *p++ << p << endl; // 435 <math>0x7ffe35b36ee4
p = genePosArray;
cout << *(++p) << p << endl; // 123 <math>0x7ffe35b36ee4
p = genePosArray;
cout << *++p << p << endl; // 123 0x7ffe35b36ee4
p = genePosArray;
cout << ++(*p) << p << endl; // 436 x7ffe35b36ee0
p = genePosArray;
cout << ++*p << p << endl; // 437 <math>\times 7ffe35b36ee0
p = genePosArray;
```

cout << (*p)++ << p << endl; // 437 x7ffe35b36ee0

Data Structures

D-6:-:+:--

Address

Pointe

Declarat

Dereference

Dereferen assignmen

Uses Issues?

Pointers to pointe

Pointers and arrays

What is an array? Array and pointer indexing

Pointer arithmetic

Operator

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