

Pointers

In this section of notes you will learn about another type of variable that stores addresses rather than data

James Tam

Memory: What You Know

- Memory is similar to a series of slots each of which can store a single piece of information

1000	1004	1008	1012
100	j	4.0	
1016	1020	1024	1028
1032	1036	1040	1044
1048	1052	1056	...

James Tam

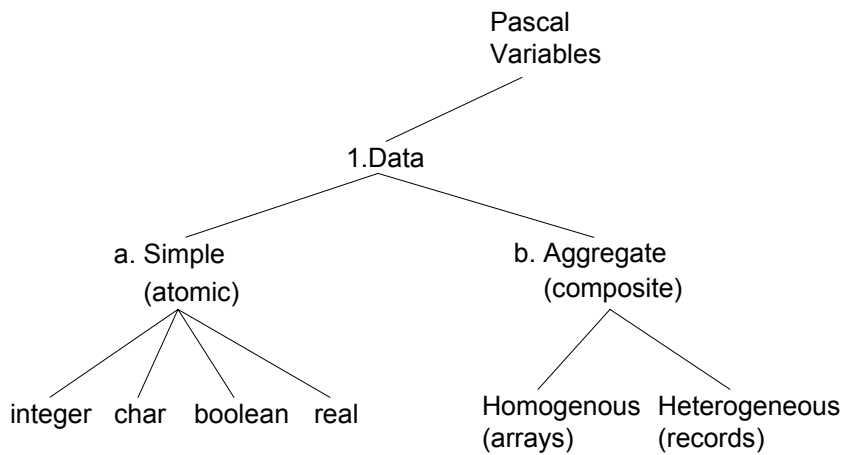
Memory: What You Will Learn

- Memory can also contain the address of another slot

1000	1004	1008	1012
100	j	4.0	1036
1016	1020	1024	1028
1032	1036	1040	1044
	t		
1048	1052	1056	...

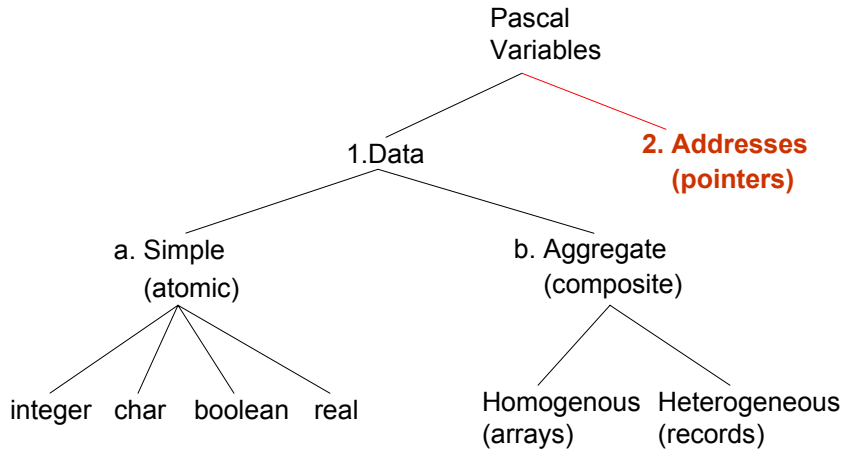
James Tam

Types Of Variables



James Tam

Types Of Variables



James Tam

Declaration Of Pointer Variables

Format:

```
type
    type name = ^ type pointed to1;
:
:
begin
    var pointer name : type name;
```

Example:

```
type
    IntegerPointer = ^integer;
:
:
begin
    var numPtr1 : IntegerPointer;
```

¹ An alternative is to use the "at-sign" @ instead of the "up-arrow" ^ to declare a pointer variable (*not recommended*)

James Tam

Allocating Memory For Pointers

Static vs. dynamic memory

- Arrays

Allocating dynamic memory

- Reserving some dynamic memory and having the pointer point to it.

Format

`new (pointer name);`

Example

`new (numPtr1);`

James Tam

De-Allocating Memory For Pointers

De-allocating memory

- Returning back the dynamically allocated memory

Format

`dispose (pointer name);`

Example

`dispose (numPtr1);`

James Tam

De-Allocating Memory For Pointers: Followup

- Should also be followed by having the pointer no longer point to the memory that has just been de-allocated

Format:

```
pointer name := NIL;
```

Example

```
numPtr1 := NIL;
```

James Tam

Using Pointers

Important! Are you dealing with the pointer or what the pointer is pointing to (allocated memory)?

- Pointer name

- Pointer name ^ (de-reference pointer)

James Tam

Using Pointers

Important! Are you dealing with the pointer or what the pointer is pointing to (allocated memory)?

- Pointer name

pointer

- Pointer name ^ (de-reference pointer)

pointer X → variable

James Tam

Accessing Pointers

Format:

(Pointer)

pointer name

(Memory pointed to)

pointer name ^

James Tam

Accessing Pointers (2)

Example:

```
type
  IntegerPointer = ^integer;
  :
  :
begin
  var numPtr1 : IntegerPointer;
  new(numPtr1);

  (Pointer)
  writeln(numPtr1);

  (Memory pointed to)
  writeln(numPtr1^);
```

James Tam

Accessing Pointers (2)

Example:

```
type
  IntegerPointer = ^integer;
  :
  :
begin
  var numPtr1 : IntegerPointer;
  new(numPtr1);

  (Pointer)
writeln(numPtr1);

  (Memory pointed to)
  writeln(numPtr1^);
```

James Tam

Using Pointers : Allowable Operations

Assignment :=

Relational

• Equality =

• Inequality \neq

James Tam

Using Pointers : Assignment

Format:

(Pointer)

pointer name := pointer name;

(Memory pointed to)

pointer name ^ := expression;

Example:

(Pointer)

numPtr1 := numPtr2;

(Memory pointed to)

numPtr1 ^ := 100;

James Tam

Using Pointers : Allowable Operations (Equality)

Format:

(Pointer)

if (*pointer name 1* = *pointer name 2*) then

(Memory pointed to)

if (*pointer name 1*^ = *pointer name 2*^) then

Example:

(Pointer)

if (numPtr1 = numPtr2) then

(Memory pointed to)

if (numPtr1^ = numPtr2^) then

James Tam

Using Pointers : Allowable Operations (Inequality)

Format:

(Pointer)

if (*pointer name 1* \diamond *pointer name 2*) then

(Memory pointed to)

if (*pointer name 1*^ \diamond *pointer name 2*^) then

Example:

(Pointer)

if (numPtr1 \diamond numPtr2) then

(Memory pointed to)

if (numPtr1^ \diamond numPtr2^) then

James Tam

Pointers : First Example

A full version of this example can be found in Unix under:
/home/231/examples/pointers/pointer1.p

```
program pointer1 (output);
type
  IntegerPointer = ^integer;
begin
  var num      : integer;
  var temp     : integer;
  var numPtr1 : IntegerPointer;
  var numPtr2 : IntegerPointer;
  writeln('Example 1');
  num := 10;
  new(numPtr1);
  new(numPtr2);
  numPtr1^ := 100;
  numPtr2^ := 100;
```

James Tam

Pointers : First Example (2)

```
writeln('num = ':11, num:3);
writeln('numPtr1^ = ':11, numPtr1^:3);
writeln('numPtr2^ = ':11, numPtr2^:3);
if (numPtr1 = numPtr2) then
  writeln('Same memory')
else
  writeln('Separate memory');
if (numPtr1 ^= numPtr2^) then
  writeln('Same data')
else
  writeln('Different data');

(* Not allowed *)
(*writeln('numPtr1=', numPtr1); *)
```

James Tam

Pointers: First Example (3)

```
writeln('Example 2');
temp := num;
num := numPtr1^;
writeln('num = ':11, num:3);
writeln('numPtr1^ = ':11, numPtr1^:3);
writeln;

writeln('Example 3');
numPtr1^ := num;
num := 2;
writeln('num = ':11, num:3);
writeln('numPtr1^ = ':11, numPtr1^:3);
writeln;
```

James Tam

Pointers: First Example (4)

```
writeln('Example 4');
numPtr2^ := 66;
numPtr1 := numPtr2;
if (numPtr1 = numPtr2) then
  writeln('Same memory')
else
  writeln('Separate memory');
numPtr2^ := 33;
writeln('numPtr1^ = ':11, numPtr1^);
writeln('numPtr2^ = ':11, numPtr2^);
dispose(numPtr1);
dispose(numPtr2);
numPtr1 := NIL;
numPtr2 := NIL;
end.
```

James Tam

Pointers As Value Parameters

Need to define a type for the pointer first!

Example (defining type)

type

```
CharPointer = ^char;
```

Format (passing pointer):

```
procedure procedure name (pointer name (1) : type of pointer (1);  
                           pointer name (2) : type of pointer (2);  
                           :  
                           :  
                           pointer name (n) : type of pointer (n));
```

```
function function name (pointer name (1) : type of pointer (1);  
                        pointer name (2) : type of pointer (2);  
                        :  
                        :  
                        pointer name (n) : type of pointer (n));
```

James Tam

Pointers As Value Parameters (2)

Example (passing pointer):

```
procedure proc1 (charPtr : CharPointer);
```

James Tam

Pointers As Variable Parameters

Need to define a type for the pointer first!

Example (defining type)

```
type
```

```
  CharPointer = ^char;
```

Format (passing pointer):

```
  procedure procedure name (var pointer name (1) : type of pointer (1);  
                             var pointer name (2) : type of pointer (2);  
                             :  
                             :  
                             var pointer name (n) : type of pointer (n));
```

Example (passing pointer):

```
  procedure proc2 (var charPtr : CharPointer);
```

James Tam

Pointers: Second Example

A full version of this program can be found in Unix under:
/home/231/examples/pointers/pointer2.p

```
program pointer2 (output);
```

```
type
```

```
  CharPointer = ^char;
```

```
procedure proc1 (charPtr : CharPointer);
```

```
var
```

```
  temp : CharPointer;
```

```
begin
```

```
  writeln;
```

```
  writeln('Proc1');
```

```
  new(temp);
```

```
  temp^ := 'A';
```

```
  charPtr := temp;
```

```
  writeln('temp^ = ', temp^);
```

```
  writeln('charPtr^ = ', charPtr^);
```

```
end;
```

James Tam

Pointers: Second Example (2)

```
procedure proc2 (var charPtr : CharPointer);
var
  temp : CharPointer;
begin
  writeln;
  writeln('Proc2');
  new(temp);
  temp^ := 'A';
  charPtr := temp;
  writeln('temp^ = ', temp^);
  writeln('charPtr^ = ', charPtr^);
end;
```

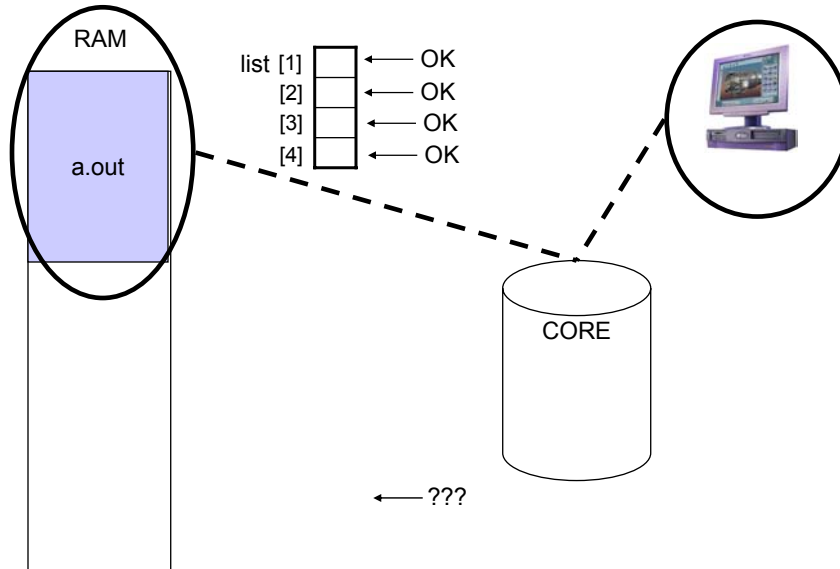
James Tam

Pointers: Second Example (4)

```
begin          (* Main program *)
  var charPtr : CharPointer;
  new (charPtr);
  charPtr^ := 'a';
  writeln;
  writeln('Main program. ');
  writeln('charPtr^ = ', charPtr^);
  proc1(charPtr);
  writeln('After proc1');
  writeln('charPtr^ = ', charPtr^);
  proc2(charPtr);
  writeln('After proc2');
  writeln('charPtr^ = ', charPtr^);
  writeln;
end.          (* End of main program *)
```

James Tam

What You Know: How Segmentation Faults Are Caused By Indexing Beyond The Array Bounds



James Tam

What You Will Learn: How Segmentation Faults (Possibly Bus Errors) Can Be Caused By Incorrect Pointer Dereferencing

A full version of this program can be found in Unix under:
`/home/231/examples/pointers/pointer3.p`

program pointer3 (output);

```
type
  IntegerPointer = ^ integer;
```

```
begin
  var numPtr1 : IntegerPointer;
  writeln('1');
  numPtr1^ := 100;
  writeln('2');
  numPtr1 := NIL;
  writeln('3');
  numPtr1^ := 100;
end.
```

James Tam

You Should Now Know

- How to declare new types that are pointers to data
- How to declare variables that are pointers
- The difference between static and dynamically allocated memory
- How to dynamically allocate memory
- How to de-allocate memory
- Why and when to set pointers to NIL
- How to access a pointer and how to access what the pointer points to
- How to assign values to a pointer and how to assign values to what the pointer points to
- What operations can be performed on pointers and how does each one work
- How to pass pointers as value and variable parameters
- How incorrect pointer usage results in problems with memory accesses