

CSE2031 Software Tools - Arrays and Pointers

Summer 2010

Przemyslaw Pawluk

Department of Computer Science and Engineering
York University
Toronto

May 18, 2010

Notes

What have we done last time?

Basic information about testing:

- Black- and Glass-box tests
- Random tests
- Regression
- Pre-, Post- and boundary conditions
- Assertions
- C directives `#declare`, `#include`
- modifiers `extern` and `static`

Notes

What we will do today?

- 1 Arrays
- 2 Pointers
- 3 Pointers and Arrays
- 4 Pointers and Functions
- 5 Multidimensional arrays, Pointers to Pointers
- 6 Useful tricks and complicated declarations

Notes

Array

- Data structure
- Grouping of data of the same type
- Indicated with brackets containing positive integer constant or expression following identifier
- Loops commonly used for manipulation
- Programmer sets size of array explicitly (static structure)

Notes

Declarations

Syntax

```
type name[size];
```

Examples

```
int bigArray[10];
double a[3];
char grade[10], oneGrade;
```

Notes

Declarations cnt.

- Declaration of the array allocates memory
char mark[6];
- Declares array of 6 integers named "mark"
- Similar to declaring five variables:
char mark[0], mark[1], mark[2], mark[3], mark[4]

Elements, indexed (subscripted) variables
Arrays are indexed from 0 to size-1!

Notes

Arrays in memory

CSE2031
Software
Tools -
Arrays and
Pointers
Przemyslaw
Pawluk

Arrays

Pointers

Pointers and Arrays

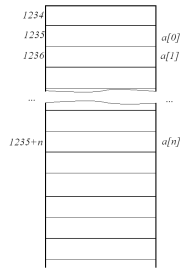
Pointers and Functions

Multidimensional

arrays,
Pointers to
Pointers

Useful
tricks and
complicated
declarations

Summary



Notes

Initialization

CSE2031
Software
Tools -
Arrays and
Pointers
Przemyslaw
Pawluk

Arrays

Pointers

Pointers and Arrays

Pointers and Functions

Multidimensional

arrays,
Pointers to
Pointers

Useful
tricks and
complicated
declarations

Summary

Initialization enclosed in curly brackets (in declaration)

```
int a[5] = {1,2};
```

Declares array a and
initializes first two elements
and all remaining set to zero

```
int b[] = {5,4,3,2,1};
```

Declares array b and
initializes all elements and
sets the length of the array
to 5

Notes

Strings=Arrays of chars

CSE2031
Software
Tools -
Arrays and
Pointers
Przemyslaw
Pawluk

Arrays

Pointers

Pointers and Arrays

Pointers and Functions

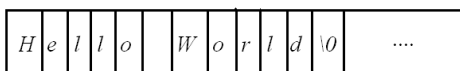
Multidimensional

arrays,
Pointers to
Pointers

Useful
tricks and
complicated
declarations

Summary

```
char msg[] = "Hello world!" ;
```



Notes

Array access

Access to elements is through the index

Read

```
x=a [ 2 ] ;
```

Write

```
a [ 2 ] = 2 ;
```

Notes

Question

What's the difference between:

```
a [ i ] ++ ;
```

```
a [ ++ i ] ;  
a [ i ++ ] ;
```

Notes

Example

Let's test the difference on the example - Write word in the column

Notes

Pointers

- Memory address of a variable
 - Declared with data type, * and identifier
- ```
type * pointerVar1 , * pointerVar2 ...;
```
- There has to be a \* before EACH of the pointer variables
  - Example.
- ```
double * p;  
int *p1 , *p2;
```

Notes

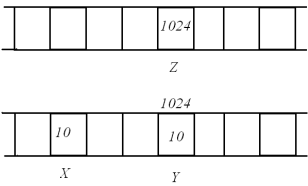
Pointers and Variables

- We can get the variable's address (pointer) using '&'
- ```
type *pointer_name = &variable ;
```
- We can get the value under the address using '\*'
- ```
type variable = *pointer_variable ;
```

Notes

Pointer Variables

```
int x , y * z ;  
  
x = 10 ;  
y = x ;  
z = &y ;
```



Notes

Be careful!

When assigning values to the pointer variable we should use '&'.
Assigning explicit value is a bad idea!

```
int *x, z;  
  
x=0x12345A;  
  
x=&z;
```

Explicit assignment is bad
idea!

'&' gives you a makes you
sure that this address exists
and is valid.

Notes

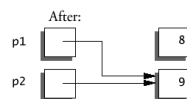
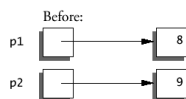
Example

Simple example with pointers

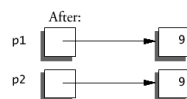
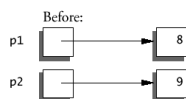
Notes

Assignments

p1 = p2;



*p1 = *p2;



Notes

Pointers and Arrays

Identifier of an array is equivalent to the address of its first element

```
int numbers [20];  
int * p;  
p = numbers; /* Valid */  
numbers = p; /* Invalid */
```

- p and numbers are equivalent and they have the same properties
- Only difference is that we could assign another value to the pointer p whereas numbers will always point to the first of the 20 integer numbers of type int

Notes

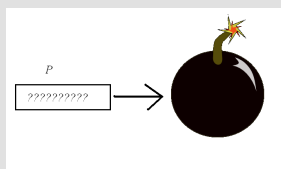
Bad pointers!

Bad Pointer

It is a pointer variable that was not initialized (has undetermined value)

What happens at runtime when the bad pointer is dereferenced?

```
int* p; /* allocate the pointer,  
        * but not the pointee */  
*p = 42; /* this dereference is a  
        * serious runtime error */
```



Notes

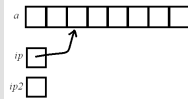
Example

Bad pointers – segmentation fault

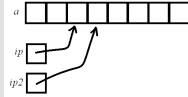
Notes

Pointer Arithmetic

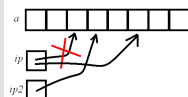
```
int *ip, *ip2;  
int a[10];  
ip = &a[3];
```



```
ip2 = ip + 1;
```



```
ip += 3;
```



Notes

Arithmetic cont.

```
ip2 = ip1 + 3;
```

```
ip2 - ip1 = 3
```

```
int array1[10], array2[10];  
int *ip1, *ip2 = array2;  
int *ep = array1;  
for(ip1 = array1; ip1 < ep; ip1++)  
    *ip2++ = *ip1;
```

Notes

Example

Pointer arithmetic – copy arrays

Notes

Void*

Generic pointer

void * is a generic pointer. Any pointer can be cast to void * and back again without loss of information

Notes

Pointer Arithmetic Summary

- void * (pointer to a void) is the generic pointer (replacing char *)
- Legal: add/sub a pointer and an integer, subtracting and comparing 2 pointers to members of the same array, and assigning or comparing to zero.
- Illegal add, multiply or divide 2 pointers, or assign one type to another type except void * without a cast.
- Any pointer can be cast to void * and back again without loss of information (used for pointer argument).

Notes

Functions

- Arrays passed to a functions are passed by reference.
- The name of the array is a pointer to its first element
- strcpy(char dest[], char src[]);
- Note that does not copy the array in the function call, just a reference to it.

Notes

Deep copy vs. Shallow copy

```
int a[3]={1,2,3}, *ip;
ip = a;
```



```
int a[3]={1,2,3}, ip[3];
ip[0] = a[0];
ip[1] = a[1];
ip[2] = a[2];
```



Notes

Multidimensional arrays

Declaration

```
int a[3][3];
```

Declaration+Initialization

```
int a[][3] = {
    {1,2,3},
    {4,5,6},
    {7,8,9}};
```

Declaration+Initialization

```
int a[3][3] = {
    {1,2,3},
    {4,5,6},
    {7,8,9}};
```

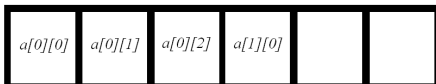
Wrong!!!

```
int a[][] = {
    {1,2,3},
    {4,5,6},
    {7,8,9}};
```

Notes

Multidimensional Arrays

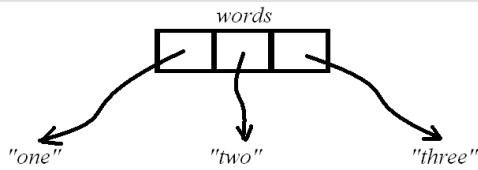
- Multi-dimensional arrays are array of arrays
- For the previous example, `m[0]` is a pointer to the first row.



Notes

Array of pointers

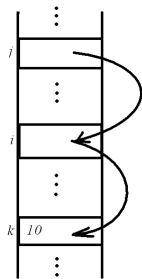
```
char *words[]={"one", "two", "three"};
```



Notes

Pointers to Pointers

```
int **j;  
int *i;  
int k=10;  
i=&k;  
j=&i;
```



Notes

Pointers vs. Arrays What's the difference?

Array of pointers

```
char *words[]={"one", "two", "three"};
```

Array of arrays

```
char words[][10]={"one", "two", "three"};
```

Notes

Pointer to Whole Array

```
Char (*p2)[100];
char name[100];
char *p1;
p1=name;
p2=name; /* What is the difference?
          * Consider p1+1 and p2+1*/
```

Notes

int argc, char*argv[]

```
main(int argc, char*argv[])
```

- argc is the number of arguments
- argv is a pointer to the array containing the arguments.
- argv[0] is a pointer to a string with the program name

Notes

Example

Print the commandline arguments

Notes

Pointers to functions

- It is possible to assign a pointer to a function.
- That pointer could be manipulated, assigned, placed on arrays, or passed/returned to/by functions.

Notes

Pointers to functions cnt.

```
int comp(void *, void *)
```

comp is a function
that has two void* arguments
and returns int

```
int (*comp)(void *, void *)
```

comp is a pointer to a
function
that has 2 void * arguments
and returns an int

Notes

Difficult Declaration

```
int *f( );
```

```
int (*daytab)[13]
```

```
int (*pf)()
```

```
char ((*x())[ ]) ( )
```

```
char **argv
```

```
char ((*x[3]) ( ) ) [5]
```

Notes

What have we done today?

- 1 Arrays
- 2 Pointers
- 3 Pointers and Arrays
- 4 Pointers and Functions
- 5 Multidimensional arrays, Pointers to Pointers
- 6 Useful tricks and complicated declarations

Notes

Notes

Notes
