



CRICOS PROVIDER 00123M

School of Computer Science

# COMP SCI 1103/2103 Algorithm Design & Data Structure Review of Pointers

adelaide.edu.au

*seek* LIGHT

# How to Submit your Practicals

- Go to <https://cs.adelaide.edu.au/services/websubmission/>
- There, find the directory name that you need to make in your svn repository for each assignment
- Use secure shell to have external access to university's computing resources:  
ssh uss.cs.adelaide.edu.au
- Use svn commands to make the directory
- Upload (or make) your files in your that directory.
- Make sure your files are compiled with university's compiler
- Go back to  
<https://cs.adelaide.edu.au/services/websubmission/> and submit your work. (**Don't forget this!**) and check feedbacks
- Get help from
  - your practical supervisors
  - <https://cs.adelaide.edu.au/current-students/handbook/external/>

# SVN

- You know what exactly the repository address of your assignment needs to be (from web-submission)
- Use svn commands to make that folder
  - `svn mkdir --parents https://version-control.adelaide.edu.au/svn /aXXXXXXX/what/ever/you/want -m "must have some comments"`
    - Either from your own terminal
    - Or from the server, which you connect to by ssh. If you have windows, use Putty (a free ssh client) to connect to the server through ssh
- If you don't have a working copy, check out this folder (or one of the parents) to get a fresh working copy of that
  - `svn co https://version-control.adelaide.edu.au/svn /aXXXXXXX/what/ever/you/want localFolderAddress`
    - Either on your own computer
    - Or on the server (first use ssh to connect)

# SVN

- If you already have a working copy of one of the parent folders,
  - Either make the assignment folder(s) locally and add them by svn commands
    - cd ExistingParentFolder
    - svn add NewFolderName
    - svn ci –m “comments”
  - Or use svn commands to make that folder(s) on the server
    - Then update your working copy folder. You will see the new directory in that
      - cd ExistingParentFolder
      - svn update
- Try not to have more than one working copy.

# Code on the server

- If your working copy is on the server
  - ssh to server each time you want to work on it
  - Go to your working copy folder
  - Use vim (or any other terminal-based text editor):  
  vim main.cpp
  - g++ -Wall -O2 \*.cpp
  - ./a.out

# Course Outline

- Week 1-5: Review, ADTs, Inheritance, Design, Recursion, Polymorphism.
- Week 6-12: Complexity, Sorting and Searching, Tree and Graph, Algorithmic strategies.

## Assumed Knowledge

You can design and build simple object-oriented programs in C++, including classes and memory management.

# Review of Pointers

In this lecture, we are going to:

- Review the concepts of pointers
- Talk about pointer arithmetic
- Discuss arrays and pointers

# Review of Pointers

- What are pointer variables?
  - A pointer variable is a variable pointing to the memory address of another variable. It gives us more control on the computer's memory.
- How do we create them?
  - You can create a pointer variable by using the `*`

```
double *ptr1, ptr2;  
double* ptr1, ptr2;  
double *ptr1, *ptr2;
```

```
typedef double* DoublePtr;  
DoublePtr ptr1, ptr2;
```

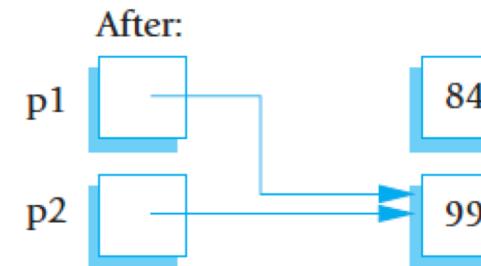
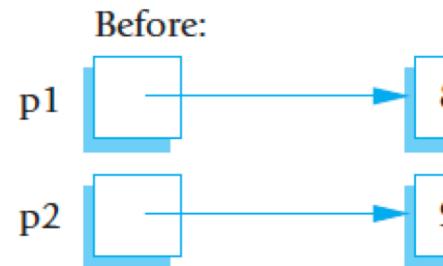
- How do we use them in C++?
  - You can refer to the address of a variable using `&`

```
double *p, v;  
p = &v;  
*p = 100;  
  
v = ?
```

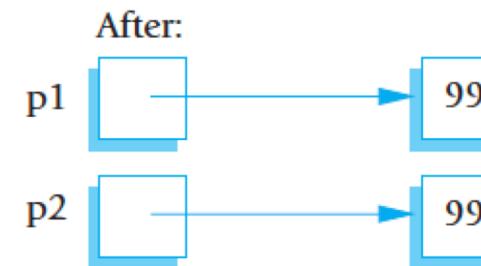
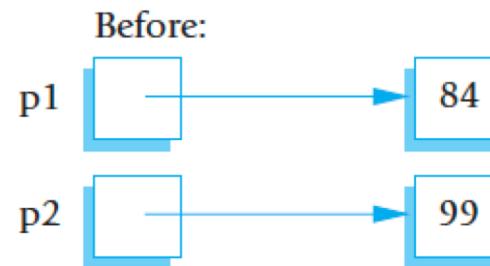
# Dereferencing Operator

- The variable pointed to by  $ptr \rightarrow *ptr$ .
- What's the difference between  $ptr1 = ptr2$  and  $*ptr1 = *ptr2$ ?

$p1 = p2;$



$*p1 = *p2;$



## Example 2

```
1 main()
2 {char c='c';
3 char* cp=&c;
4 int i=100;
5 int* ip=&i;
6 int j=*ip;
7 }
```

Questions:

1. \*ip = 99; i = ?
2. \*cp = \*cp + 1; c = ?
3. &ip = ?

Addr	Name	Value
0xbffff374	i	100
0xbffff375		
0xbffff376		
0xbffff377		
0xbffff378	?	?
0xbffff379		
0xbffff37a		
0xbffff37b	c	'c'
0xbffff37c	cp	0xbffff37b
0xbffff37d		
0xbffff37e		
0xbffff37f		
0xbffff380	ip	0xbffff374
0xbffff381		
0xbffff382		
0xbffff383		
0xbffff384	j	100
0xbffff385		
0xbffff386		
0xbffff387		

# Why Pointers/references?

- Instead of passing large quantities of data between functions, we can
  - just pass the pointer to the start of the data
  - saves memory and data transfer
- Dynamic memory usage
- Pass parameters to a function by reference

# Pass-by-reference

```
void swapByValue(int x, int y){  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}
```

```
void swapByPointer(int* x, int* y){  
    int temp;  
    temp = *x;  
    *x = *y;  
    *y = temp;  
}
```

# Dynamic Variables

- What are dynamic variables?
  - Variables that are created by using *new* operator.
  - They are created and destroyed while the program is running.

```
1 //Program to demonstrate pointers and dynamic variables.  
2 #include <iostream>  
3 using namespace std;  
4  
5 int main()  
6 {  
7     int *p1, *p2;  
8  
9     p1 = new int;  
10    *p1 = 42;  
11    p2 = p1;  
12    cout << "*p1 == " << *p1 << endl;  
13    cout << "*p2 == " << *p2 << endl;  
14    *p2 = 53;  
15    cout << "*p1 == " << *p1 << endl;  
16    cout << "*p2 == " << *p2 << endl;  
17    p1 = new int;  
18    *p1 = 88;  
19    cout << "*p1 == " << *p1 << endl;  
20    cout << "*p2 == " << *p2 << endl;  
21    cout << "Hope you got the point of this example!\n";  
22    return 0;  
23 }
```

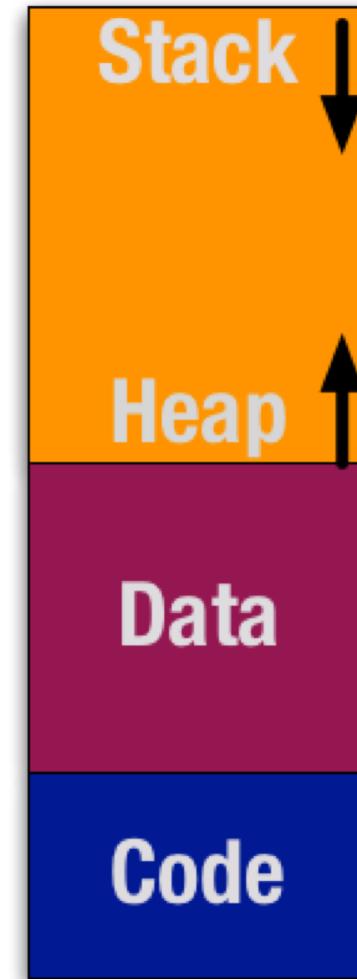
\*p1 == 42  
\*p2 == 42  
\*p1 == 53  
\*p2 == 53  
\*p1 == 88  
\*p2 == 53

# Dynamic Variables

- “new” allocates a part of memory as the dynamic variable.
- We need to “delete” dynamic variables after use, to prevent memory leak
  - type \* ptr= new type;
  - Set a value to \*ptr and use it
  - When you no longer need it:  
    delete ptr;
- The C++ standard specifies that if there is no sufficient memory available to create the new variable -> the new operator, by default, terminates the program.

# Where are the variables stored?

- Static variables and parameters are stored in Stack.
- The rest of the allocated memory is used as the heap; for dynamic memory allocation
- The stack and heap are in a shared area.
- The memory that is allocated to variables and parameters from stack is not released until their function is returned. Unlike memory allocation in Heap.



# Pointer Arithmetic

- You cannot perform the normal arithmetic operations on pointers.
  - multiplication or division is not allowed
  - Addition and subtraction are different
- The pointer arithmetic depends on the size of the type that pointer is of that type.
- Pointer arithmetic (+/-) shifts the address by a number of bytes equal to the size of the pointer type

```
int i=100;  
int* ip=&i;
```

Addr	Name	Value
0xbffff380	ip	0xbffff374
0xbffff381		
0xbffff382		
0xbffff383		

What is the value of ip+1?  
0xbffff378

# Pointers and Arrays

- In C++, an array variable is actually a pointer variable that points to the first indexed variable of the array.

```
int *ptr;
int a[10];
int i;

for(i = 0; i<10; i++){
    a[i] = i*2;
}

ptr = a;

for(i = 0; i<10; i++){
    cout << ptr[i] << " ";
}
cout << endl;

ptr[5] = 5;

for(i = 0; i<10; i++){
    cout << a[i] << " ";
}
cout << endl;
```

By the way! You can go beyond the size of the array

0 2 4 6 8 10 12 14 16 18

Iterating through ptr is the same  
as iterating through array a.

0 2 4 6 8 5 12 14 16 18

# C-string

- A pointer-based string in C++ is an array of characters ending with the null terminator ('\0').
- The null terminator indicates where a string terminates in memory.
- A C-string can be accessed via a pointer

```
char city[9] = "Adelaide";
char *ptrCity = "Adelaide";

char city[] = {'A', 'd', 'e', 'l', 'a', 'i', 'd', 'e'};    not equivalent!
```

```
cout << city[1] << endl;
cout << *(city+1) << endl;
cout << ptrCity[1] << endl;
cout << *(ptrCity+1) << endl;
```

All output the second element of the string.

# Summary

- Pointers allow you to access storage but it's extremely manual. Misjudging your pointer arithmetic will have strange results.
- Space is finite - management is important.
- C++ arrays and pointers are very easy to cause problems!



THE UNIVERSITY  
ofADELAIDE



# Questions?