CSC180: Lecture 16

Wael Aboulsaadat

wael@cs.toronto.edu

http://portal.utoronto.ca/

Acknowledgement: These slides are partially based on the slides supplied with Prof. Savitch book: Problem Solving with C

Pitfall: Stack Overflow

- Because each recursive call causes an activation frame to be placed on the stack
  - infinite recursion can force the stack to grow beyond its limits to accommodate all the activation frames required
  - The result is a stack overflow
  - A stack overflow causes abnormal termination of the program

Recursion Types

- Recursion for Tasks
  - E.g. binary search, sorting (later…)
- Recursion for Values
  - E.g. power, factorial, etc…

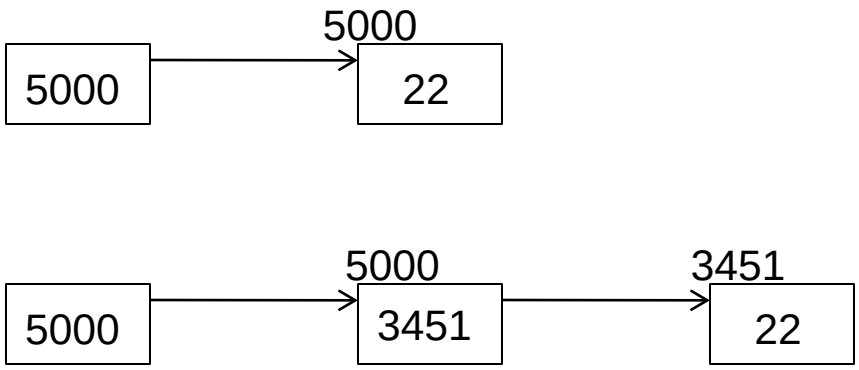
Recursion versus Iteration

- Any task that can be accomplished using recursion can also be done without recursion
  - A nonrecursive version of a function typically contains a loop or loops
  - A non-recursive version of a function is usually called an iterative-version
  - A recursive version of a function
    - Usually runs slower
    - Uses more storage
    - May use code that is easier to write and understand

Pointers

Pointers intro

- a memory address of a computer which may contain other variable or even another pointer



Pointer variable

- *General form of pointer declaration is:*

```
type   *name;
```

E.g.

```
int     *pnValue;
float   *pfValue;
char    *pcValue;

int     *pnValue, *pnIndex;
```

Pointers initialization

- You can initialize a pointer to null using 2 methods as shown below -

```
variable_type *pointer_name = 0;
```

or

```
variable_type *pointer_name = NULL;
```

NULL is defined in many standard headers such as <stdio.h>.

Pointer assignment

- assign value of one pointer to another using assignment operator '='

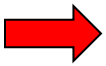
- right hand side points to memory address of variable stored in left hand side pointer. As a result both pointers point to same memory location

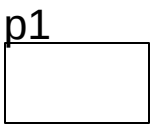
Pointer assignment - example

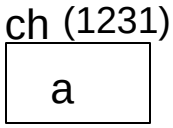
```
#include <stdio.h>

int main ()
{
   char ch = a;
   char* p1, *p2;

   p1 = &ch;

   p2 = p1; // Pointer Assignment Taking Place

   printf (" *p1 = %c And *p2 = %c", *p1,*p2); // Prints 'a' twice

   return 0;
}
```

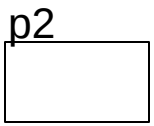
Pointer assignment - example

```
#include <stdio.h>

int main ()
{
   char ch = a;
   char* p1, *p2;

   p1 = &ch;

   p2 = p1; // Pointer Assignment Taking Place

   printf (" *p1 = %c And *p2 = %c", *p1,*p2); // Prints 'a' twice

   return 0;
}
```



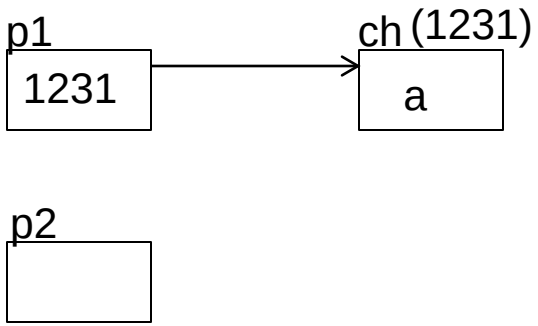






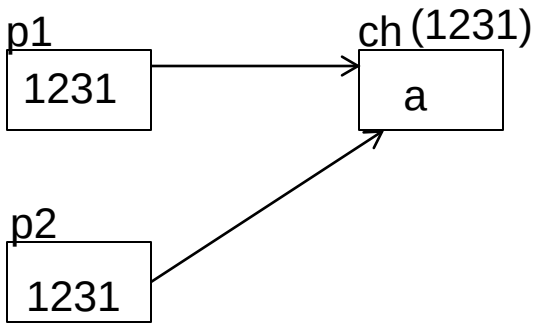
Pointer assignment - example

```
#include <stdio.h>

int main ()
{
   char ch = a;
   char* p1, *p2;

   p1 = &ch;

   p2 = p1; // Pointer Assignment Taking Place

   printf (" *p1 = %c And *p2 = %c", *p1,*p2); // Prints 'a' twice

   return 0;
}
```



Pointer assignment - example

```
#include <stdio.h>

int main ()
{
   char ch = a;
   char* p1, *p2;

   p1 = &ch;

   p2 = p1; // Pointer Assignment Taking Place

   printf (" *p1 = %c And *p2 = %c", *p1,*p2); // Prints 'a' twice

   return 0;
}
```



Pointer conversion

- Pointer conversion involves changing the type the pointer is pointing to

- Null pointer: *a pointer which points to nothing.*
  - *Infact it points to the base address of your CPU registers and since register is not addressable ➔ will lead to crash or at minimum a segmentation fault.*

- Void pointer: *technically is a pointer which is pointing to the unknown.*
  - Void pointer has special property that it can be type casted (i.e. change type) into any other pointer

Pointer conversion - example

```
#include <stdio.h>

int main ()
{
   int      i = 10;
   char  *p1
   int     *p2;

   p2 = &i;

   p1 = (char *) p2; // Type Casting and Pointer Conversion

   printf (" *p1 = %c And *p2 = %d", *p1,*p2);

   return 0;
}
```

Pointers arithmetic

- Not all arithmetic operations are defined in pointers.
  - You can increment them,
  - You can decrement them,
  - You can add and subtract integer values from them.
  - You even can subtract two pointers.

  - But you cannot add two pointers, mulitply, divide,modulus them. You can not also add or subtract values other than integer.

Pointers arithmetic - examples

- if X pointer is char type(assumed 1 Byte or 8Bit long) than X = X + 1 will have value 1001 and X = X - 1 will have value 999.
- if X pointer is integer type (assumed 2 byte or 32 bit long) than X = X + 1 will have value 1002 and X = X - 1 will have value 998.
- if X pointer is float type (assumed 4 Byte or 32Bit long) than X = X + 1 will have value 1004 and X = X - 1 will have value 9996.
- Reason: when you increment a pointer of certain base type it increase it value in such a way that it points to next element of its base type. If you decrement a pointer its value decrease in such a way that it points to previous value of its base type. So increment as well as decrement in fixed quanta of size of the base type.