GENERIC PROGRAMMING

Generic Pointers (void *)

Function Pointers

Generic Programming

- C++ has Templates
- C has void* and function pointers
- How do we write a Linked List that accepts **any** data type?
- How do we write a Linked List that accepts **any** data type and keeps the list in **sorted** order?
- How do we apply the same function to every element in a list?   Print?

Generic Programming

- Abstract data types that can store multiple types of data
- Functions that can work on multiple types of data

Generic Pointer (void*)

- void *pPtr;
- Void pointers refer to a generic untyped location on memory
  - They have no type!
- You must cast a void * to a typed pointer before using it
- Void pointers cannot be dereferenced or used in pointer arithmetic

void * Example

```
int main()
{
  int i;
  char c;
  void *the_data;

  i = 6;
  c = 'a';

  the_data = &i;
  printf("the_data points to the integer value %d\n",
          *(int*) the_data);

  the_data = &c;
  printf("the_data now points to the character %c\n",
          *(char*) the_data);

  return 0;
}
```

void*

```
typedef struct Node* NodePtr;

typedef struct Node
{
  void* pData;
  unsigned int size;
  NodePtr *psNext;
} Node;

NodePtr pNode;
                    // what value does sizeof
return?
pNode = (NodePtr) malloc(sizeof(Node));
```

The list does not know what *type* **`data`** points to.

Let's write lstInsert()

Writing lstInsert

```
ERROR_CODE lstInsert(void *pData, int size, NodePtr *hList)
```

Calling lstInsert

- Write the code necessary to insert an integer into a linked list

Compare

size_t
unsigned ??
in stdlib.h via stddef.h

- If we insert two ints into a linked list, how do we compare them?

- How do we compare two void* items?
  - no data type information

```
#include <string.h>
int memcmp(void* ptr, void*  ptr2, size_t size);
void* memcpy(void* dest, void* src, size_t size);
```

size_t

- Look in a C Eclipse Project | Includes | /usr/lib64/gcc/x86_64-suse-linux/4.5/include | stddef.h
- line 208

```
#define __SIZE_TYPE__ long unsigned int

typedef __SIZE_TYPE__ size_t;
```

Function Pointers

```
returnType (*name)(paramType ...)
```

- Example:

```
 int (*foo)(int);

 int negate(int x)
 {
   return -x;
 }

 foo = &negate;
(*foo)(3);
```

Inserting into a Sorted List

- How can we tell which node is larger?
- If the node is storing primitive data types, then we can use a function:

```
int isGreaterThan(const void *pLeft,
                  const void *pRight)
```

Inserting into a Sorted List

```
ERROR_CODE lstInsertSorted(ListPtr psList,
                                  void* pData,
                             unsigned int size,
int (*compare)(const void*, const void*));
```

Can we print every element of the list?

- We know the data type stored in the void*

  The List does not know the data type!

```
// assume we have ints in the list
void print(void* pData)
{
    printf("%d ", * ((int*) pData) );
}
```