Generic Programming

Two components

- Data

- Behavior/Functions

Linked List with int

```
typedef struct ListElement
{
  int data;
  struct ListElement *psNext;
} ListElement;


void printList(ListElement *psList)
{
  ListElement *psTemp = psList;

  while( psTemp )
  {
    printInt(psTemp->data);
    psTemp = psTemp->psNext;
  }
}
```

Linked List with void*

```
typedef struct ListElement
{
  void *psData;
  struct ListElement *psNext;
} ListElement;


void printList(ListElement *psList)
{
  ListElement *psTemp = psList;

  while( psTemp )
  {
    // printInt(psTemp->psData); ????


    psTemp = psTemp->psNext;
  }
}
```

Function Pointers!

- Each function is stored at an address.
- Pointers are just an address
- We can have pointers to functions!

- Function pointers can be passed as arguments to other functions or return from functions

- Define the function pointer
  - **`returnType (*name)(paramType ...)`**

Linked List with void*

```
typedef struct ListElement
{
  void *psData;
  struct ListElement *psNext;
} ListElement;
```

```
void printList(ListElement *psList,
{
  ListElement *psTemp = psList;

  while( psTemp )
  {
    // printInt(psTemp->psData); ????


    psTemp = psTemp->psNext;
  }
}
```

print functions

Design

- Separate data from structure
  - one function walks the linked list
    - always operates the same way
  - another function is used to interact with the data in each element

Other functionality

Storing Functions

```
typedef void(*DataVisitor)(void*);

typedef struct ListElement
{
  void *psData;
  struct ListElement *psNext;
} ListElement;

typedef struct List
{
  ListElement *psHead;
  DataVisitor pPrinter;
  DataVisitor pInitializer;
  DataVisitor pUpdater;
} List;
```

Classic Example: Sorting

```
void bubbleSort(int aInts[], int size)
{
  bool bExchange = true;

  int indx;
  int temp;

  while( bExchange )
  {
    bExchange = false;

    for (indx = 0; indx < size - 1; ++ indx)
    {
      if( aInts[indx] > aInts[indx + 1] )
      {
         temp = aInts[indx];
         aInts[indx] = aInts[indx + 1];
         aInts[indx + 1] = temp;
         bExchange = true;
      }
   }
}
```

Void * array

- How do we access elements in an array pointed to by a void*?

  int aInts[10];

  void* pArray = aInts;

  // print aInts[2]

  printf(“%d”,

Classic Example: Sorting

```
void genericBubbleSort(void* pArray, int elementSize, int size,
                                                    // greaterThan
)
{
  bool bExchange = true;

  int indx, temp;

  while( bExchange )
  {
    bExchange = false;

    for (indx = 0; indx < size - 1; ++ indx)
    {
      if( (*greaterThan)((void*) ((char*) pArray + indx * elementSize),
                (void*) ((char*) pArray + (indx + 1) * elementSize)
         )
      {

         bExchange = true;
      }
   }
}
```

greaterThan for ints

structs

```
typedef struct Person
{
  int ID;
  char szFName[100];
  char szLName[100];
} Person;


int personIDGreater(void* pLeft, void *pRight);
int personIDGreater(void* pLeft, void *pRight);

int personLNameGreater(void* pLeft, void *pRight);
int personLNameGreater(void* pLeft, void *pRight);
```