Lecture 3.2

Pointers (cont.)

Copying using character pointers

```
#include <stdio.h>
int main()
{
    char str1[] = "Hello World";
    char str2[] = "Goodbye World";
    char *cpt1;
    char *cpt2;
    cpt1 = &str1[0];
    cpt2 = &str2[0];
```

Copying using character pointers (cont)

```
printf("str1 is %s\n",str1);
printf("str2 is %s\n",str2);
printf("cpt1 is %s\n",cpt1);
printf("cpt2 is %s\n",cpt2);
cpt2 = cpt1;
printf("str1 is %s\n",str1);
printf("str2 is %s\n",str2);
printf("cpt1 is %s\n",cpt1);
printf("cpt2 is %s\n",cpt2);
```

Copying using character pointers (cont2)

Results:

str1 is Hello World
str2 is Goodbye World
cpt1 is Hello World
cpt2 is Goodbye World
str1 is Hello World //contents are the same
str2 is Goodbye World //contents are the same
cpt1 is Hello World
cpt2 is Hello World

Point to note:

- Actually string hasn’t been copied through the use of pointers. (why?????)
- Copy means copy the content and paste into different location.
- Here we are not coping the content just changing the pointer reference from one location to another.

Copying element by element

```
#include <stdio.h>
int main()
{
    int i;
    char str1[] = "Hello World";
    char str2[] = "Goodbye World";
    printf("str1 is %s\n",str1);
    printf("str2 is %s\n",str2);
    i = 0;
    while ((str2[i] = str1[i]) != '\0') {
      i++;
    }
```

Copying element by element (cont.)

```
    …
    printf("str1 is %s\n",str1);
    printf("str2 is %s\n",str2);
    return 0;
}
```

Copying element by element (cont.)

Results:

```
str1 is Hello World
str2 is Goodbye World
str1 is Hello World
str2 is Hello World
```

Practice Problem 2

- Try reimplementing the above program using pointers in the copy loop.
- Hints:

```
cpt1 = &str1[0];
cpt2 = &str2[0];
```

Use these pointers in the while loop, remember to dereference.

Copying element by element using pointers

```
#include <stdio.h>
int main()
{
   char str1[] = "Hello World";
   char str2[] = "Goodbye World";
   char *cpt1;
   char *cpt2;
```

Copying element by element using pointers (cont.)

```
cpt1 = &str1[0];
cpt2 = &str2[0];
printf("str1 is %s\n",str1);
printf("str2 is %s\n",str2);
while ((*cpt2 = *cpt1) != '\0') {
    cpt2++;
    cpt1++;
}
```

Code comparison

-
  ```
  while ((*cpt2 = *cpt1) != '\0') {
  cpt2++;
  cpt1++;
  ```

-
  ```
  i = 0;
  while ((str2[i] = str1[i]) != '\0') {
       i++; }
  ```

-
  ```
  while (str1[i] != ‘\0’){
       str2[i]=str1[i];
       i++; }
  ```

Calling functions by reference

- Passing a parameter by reference means that we are “referencing” the parameter, **not** passing the **value** of the parameter.
- There is two main reasons to use pass by reference:
  - To allow functions to modify several values at a time.
  - To pass large data objects to a function and avoid the overhead of copying.

Call by value

```
#include <iostream>
int cubeByValue (int); //prototype
int main( ){
   int number = 5;
   . . .
   number = cubeByValue (number);
   . . .
}
int cubeByValue (int n)
{ return n * n * n; }
```

Call by reference

```
#include <iostream>
void cubeByReference (int *); //prototype
int main( ){
   int number = 5;
   . . .
   cubeByReference (&number);
   . . .
}
void cubeByReference (int *nPtr)
{ *nPtr = *nPtr * *nPtr * *nPtr; }
```

Arrays of Pointers

- Arrays may contain pointers.
- The most common use is to form an array of strings
- char *names[4] = {“Atish”,”Itendra”,
  ”Ron”,”Vilimone”}

  names[0] ‘A’ ‘t’ ‘i’ ‘s’ ‘h’ ‘\0’
  names[1] ‘I’ ‘t’ ‘e’‘n’ ‘d’ ‘r’ ‘a’ ‘\0’
  names[2] ‘R’ ‘o’‘n’‘\0’
  names[3] ‘V’ ‘i’ ‘l’ ‘i’ ‘m’ ‘o’ ‘n’ ‘e’ ‘\0’
- char [4][9]

“sizeof” operator

- **Definition:** The sizeof operator returns the size of an object in bytes as a value of type size_t, which is usually unsigned int.
- Example

```
size_t mySize;
int x;
mySize = sizeof (x);
```

sizeof examples

```
int myInt;
printf("Size of int is %d\n",sizeof(myInt)); //argument of sizeof is an
                                              //object
printf("Size of int is %d\n",sizeof(int)); //argument of sizeof is a
                                              //data type
printf("Size of char is %d\n", sizeof(char));
printf("Size of short is %d\n", sizeof(short));
printf("Size of int is %d\n", sizeof(int));
printf("Size of long is %d\n", sizeof(long));
printf("Size of float is %d\n", sizeof(float));
printf("Size of double is %d\n",sizeof(double));
```

sizeof and Objects

- **Examples:**

```
sizeof(float);
float value;
sizeof(value);
class Cat {
   ...
};
sizeof(Cat);
```