Instructor created

6.24 Functions with pointer parameters

https://www.tutorialspoint.com/cplusplus/cpp_passing_pointers_to_functions.htm

C++ allows you to pass a pointer to a function. To do so, simply declare the function parameter as a pointer type.

Following a simple example where we pass an unsigned long pointer to a function and change the value inside the function which reflects in the calling function –

Code Editor 6.24.1: Pass a pointer to a function

Click run.

Load default template...

```
#include <iostream>
#include <ctime>

using namespace std;

void getSeconds(unsigned long *par) {
   // get the current number of seconds
   *par = time( NULL );

   return;
}

int main () {
   unsigned long sec;
   getSeconds( &sec );

   // print the actual value
   cout << "Number of seconds :" << sec << endl;
```

Run

The function which can accept a pointer, can also accept an array as shown in the following example –

Code Editor 6.24.2: Pass a pointer to a function

Click run.

Load default template...

```
#include <iostream>
using namespace std;


double getAverage(int *arr, int size) {
   int i, sum = 0;
   double avg;

   for (i = 0; i < size; ++i) {
      sum += arr[i];
   }
   avg = double(sum) / size;

   return avg;
}

int main () {
   // an int array with 5 elements.
```

Run

C++ allows you to return a pointer from a function. To do so, you would have to declare a function returning a pointer as in the following example -

Figure 6.24.1: Function returns a pointer

```
int* myFunction() {
   .
   .
   .
}
```

Second point to remember is that, it is not a good idea to return the address of a local variable to outside of the function, so you would have to define the local variable as dynamic allocated variable.

Now, consider the following function, which will generate 10 random numbers and return them using an array name which represents a pointer i.e., address of first array element.

Code Editor 6.24.3: Function returns a pointer

Click run.

Load default template...

```
#include <iostream>
#include <ctime>

using namespace std;

// function to generate and retrun random numbers.
int* getRandom( ) {
   int* r = new int[10];
   // set the seed
   srand( (unsigned)time( NULL ) );

   for (int i = 0; i < 10; ++i) {
      r[i] = rand();
      cout << r[i] << endl;
   }
   return r;
}
```

Run

If you want to change the address of a pointer in a function, you must use double pointers as in the following example -

Code Editor 6.24.4: Changing address of a pointer in a function

Click run.

Load default template...

```
#include <iostream>

using namespace std;

void changeAddress(int** p) {
   *p = NULL;  /* set pointer to null */
}
void notChangeAddress(int* p) {
   p = NULL;  /* makes copy of p and copy is set to null*/
}

int main() {
   int* k;
   notChangeAddress(k);   /* k unchanged */
   cout << k << endl;
   changeAddress(&k);   /* NOW k == NULL */
   cout << k << endl;
}
```

Run

Another example -

Code Editor 6.24.5: Change a pointer address in a function

Click run.

Load default template...

```
#include <iostream>

using namespace std;

// use double pointers to change the original pointer address
void changeAddress(int** p) {
   int* n = new int;
   *n = 20;
   delete *p;  // deallocate the old space
   *p = n;  /* change to the new address */
}


int main() {
   int* k = new int;  // create a pointer with allocated space
   *k = 10;  // store 10 in the allocated space
   cout << *k << endl;  // print 10 from the allocated space
   changeAddress(&k);   /* pass in the address of the pointer */
```

Run

If you want to change the address of a dynamic allocated array in a function, you can use the similar approach from the above example.

Code Editor 6.24.6: Print Array with pointer

Click run

Load default template...

```
#include <iostream>

using namespace std;

// print the array
void printArray(int* n, int size) {
   for (int i=0; i<size; i++) {
      cout << n[i] << " ";
   }
   cout << endl;
}

// change the array address with the new dynamic allocated array example
// Notes: the size is updated after the function returns
void greaterZero(int** n, int& size) {
   int count = 0;
   // count number of values greater than zero
```

Run

Feedback?