```cpp
/******************************************************************************

                              Online C++ Compiler.
               Code, Compile, Run and Debug C++ program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/

#include <iostream>
#include <cstring>

using namespace std;

int passByValue(int value) {
    value *= value;
    return value;
}

int passByReference(int * value) {
    *value *= *value;
}

int passConst(const int & value) {
    return value * value;
}

int & passRef(int & value) {
    value *= value;
    return value;
}

int * passPtr(int * value) {
    *value *= *value;
    return value;
}


int * squarePtr(int number) {
   int * dynamicAllocatedResult = new int(number * number);
   return dynamicAllocatedResult;
}

int & squareRef(int number) {
   int * dynamicAllocatedResult = new int(number * number);
   return *dynamicAllocatedResult;
}

int count(const char *str, const char c) {
   int count = 0;
   while (*str) {   // same as (*str != '\0')
```

```
      if (*str == c) ++count;
      ++str;
   }
   return count;
}

int main()
{
    cout << "---- Pointer & Refs ----" << endl;
    int n = 3;
    /* Pass-by value -> Þá sendir maður bara inn gildið */
    cout << passByValue(n) << endl; // Skilar 9

    /* Pass-by-Reference (with Pointer Arguments) ef maður vill breyt upprunalega stakin t.d arr
    passByReference(&n);
    cout << n << endl; // Skilar 9

    n = 3;
    /* "const" Function Reference/Pointer */
    cout << passConst(n) << endl;

    /* Passing the Return-value as Reference */
    int & res = passRef(n);
    cout << res << endl;  // Skilar 9

    n = 3;
    int * ptrRes = passPtr(&n);
    cout << *ptrRes << endl; // Skilar 9

    n = 3;
    /* Passing Dynamically Allocated Memory as Return Value by Reference*/
    cout << *squarePtr(n) << endl;  // 9
    n = 3;
    cout << squareRef(n) << endl;   // 9

    /*
        In pass-by-value, a clone is made and passed into the function. The caller's copy cannot
        In pass-by-reference, a pointer is passed into the function. The caller's copy could be
        In pass-by-reference with reference arguments, you use the variable name as the argument
        In pass-by-reference with pointer arguments, you need to use &varName (an address) as th
    */

    cout << "---- Pointer arithmatic ----" << endl;
    /* Pointer arithmatic */
    int numbers[] = {10,20,30,40,50};
    int * iPtr = numbers;       // Create pointer to the array
    cout << iPtr << endl;       // 0x7ffee4021710
    cout << iPtr + 1 << endl;    // 0x7ffee4021710 (increase by 4 - sizeof int)
    cout << *iPtr << endl;       // 10
    cout << *(iPtr + 1) << endl; // 20
    cout << *iPtr + 1 << endl;   // 11
```

```
/* Size of array */
cout << "---- Array Size ----" << endl;
int number[100];
cout << sizeof(number) << endl; // Size of entire array in bytes (400)
cout << sizeof(number[0]) << endl;  // Size of first element of the array in bytes (4)
cout << "Array size is " << sizeof(number) / sizeof(number[0]) << endl;  // (100) int er 4 b

/* Passing arrays in/Out of a function */
// For example, the following declarations are equivalent:

// int max(int numbers[], int size);
// int max(int *numbers, int size);
// int max(int number[50], int size);
// void fun(const int *arr, int size);
// They will be treated as int*


/*

    STRINGS

    C-string (of the C language) is a character array, terminated with a null character '\0'
*/
cout << "---- Strings ----" << endl;

char msg1[] = "Hello";
char *msg2 = "Hello";

cout << strlen(msg1) << endl;       // 5
cout << strlen(msg2) << endl;       // 5
cout << strlen("Hello") << endl;    // 5

int size = sizeof(msg1)/sizeof(char);
cout << size << endl;  // 6 - including the terminating '\0'

// Print the string
for (int i = 0; msg1[i] != '\0'; ++i) {
  cout << msg1[i];
}
cout << "" << endl;

// count occurrance in string
cout << count("Hello, world", 'l') << endl;


/*

    Function Pointers

*/
```

```
    /*
        passing function as Arguments

        // Function-pointer declaration
        return-type (* function-ptr-name) (parameter-list)

        // Examples
            double (*fp)(int, int)  // fp points to a function that takes two ints and returns a
            double *dp;             // dp points to a double (double-pointer)
            double *fun(int, int)   // fun is a function that takes two ints and returns a doubl

            double f(int, int);      // f is a function that takes two ints and returns a double
            fp = f;                 // Assign function f to fp function-pointer
    */



    return 0;
}
```