Programming For Engineers

Pointers in C Programming: Part 01

by
Wan Azhar Wan Yusoff[1], Ahmad Fakhri Ab. Nasir[2]
Faculty of Manufacturing Engineering
wazhar@ump.edu.my[1], afakhri@ump.edu.my[2]

0.0 Chapter's Information

- Expected Outcomes
  - To use pointers in C programming

- Contents
  1.0 What is a pointer?
  2.0 Pointer Arithmetic
  3.0 How to declare and use pointers?
  4.0 Pointer and function

1.0 What is a pointer?

- A pointer is a variable that stores an address value of another variable. An example:

```
int number = 56;
//This is to create a memory space named number and stores the value 56 in it.
int*ptrNumber = &number;
//This is to create a memory space named ptrNumber and stores the address of
//memory location named number.
```

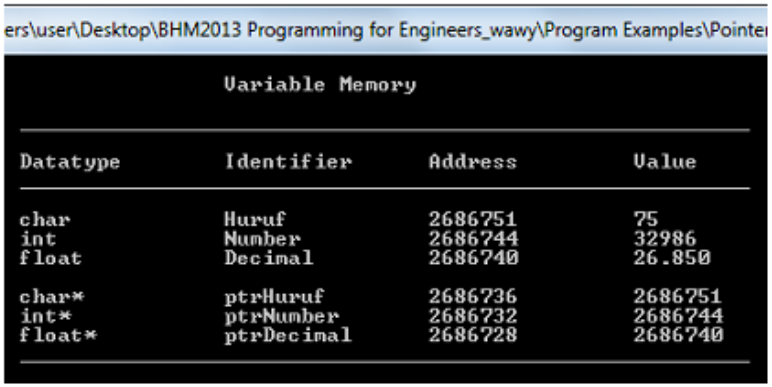
- Suppose that we create three type of variables; a character, an integer and a float. Then we can create three pointer variables; one is to store the address of the character, one is to store the address of the integer variable and one is to store the address of the float variable. The program below does this.

1.0 What is a pointer?

```c
#include <stdio.h>
/* This program is to illustrate the meaning of pointers Written by: WAWY February 2012 */
int main()
{
    /*   Create the standard data type */
    char Huruf = 'K'; //Huruf is a Melayu language word which means character
    int Number = 32986;
    float Decimal = 26.85;
    /*      This following is how we declare and initialize pointers
            Notice that the prefix "ptr" is just a convention so that
            we can recognize the pointer variables easily. */
    char* ptrHuruf = &Huruf;
    int* ptrNumber = &Number;
    float* ptrDecimal = &Decimal;
    /*      Display the variable addresses and contents */
    printf("\n\t\t\tVariable Memory\n\n");
    printf("\t_____\n\n");
    printf("\tDatatype\tIdentifier\tAddress\t\tValue\n");
    printf("\t_____\n\n");
    printf("\tchar\t\tHuruf\t\t%u\t\t%d\n",&Huruf,Huruf);
    printf("\tint\t\tNumber\t\t%u\t\t%d\n",&Number,Number);
    printf("\tfloat\t\tDecimal\t\t%u\t\t%6.3f\n",&Decimal,Decimal);
    printf("\n"); printf("\tchar*\t\tptrHuruf\t%u\t\t%d\n",&ptrHuruf,ptrHuruf);
    printf("\tint*\t\tptrNumber\t%u\t\t%d\n",&ptrNumber,ptrNumber);
    printf("\tfloat*\t\tptrDecimal\t%u\t\t%d\n",&ptrDecimal,ptrDecimal);
    printf("\t_____\n\n");
    /*  End the program. */
    return 0;
}
```

1.0 What is a pointer?

- The output is given below.



- Notice that the contents/values of the three pointers are the addresses of the variables. For example, Huruf is a char variable that stores character constant 'K' which is equivalent to 75 in integer value. Now, where this value is stored? It is stored at the memory address 2293535. Meanwhile, ptrHuruf is declared as the pointer to variable Huruf. It stores the address of Huruf which is 2293535.

1.0 What is a pointer?

- You can also check the values of the other two pointer variables.
- To recap, A POINTER IS A VARIABLE THAT STORES THE ADDRESS VALUE OF A VARIABLE.
- What data type is a pointer? It is a 4-byte unsigned integer. That means that the address range is from 0 to 232 – 1. Do you know the value of 232 -1? Punch your calculator. It is 4,294,967,295. This number is called 4 Gig. Thus, our memory is four-gig[1].

---

[1] Certainly, it is not Ryan Gig! When a size is 2^{10}, we call it kilo byte. But the actual bytes are 1024 bytes not exactly 1000 bytes that we expect. Similarly, 2^{20}, is called mega-bytes. The actual bytes are 1,048,576. For giga, the bytes are 2^{30}.

2.0 Pointer Arithmetic

- Consider the normal integer variable a,

```
int a = 13;
a = a+1;
```

- Obviously, the value of variable a+1 is 14. But consider the pointer ptrA;

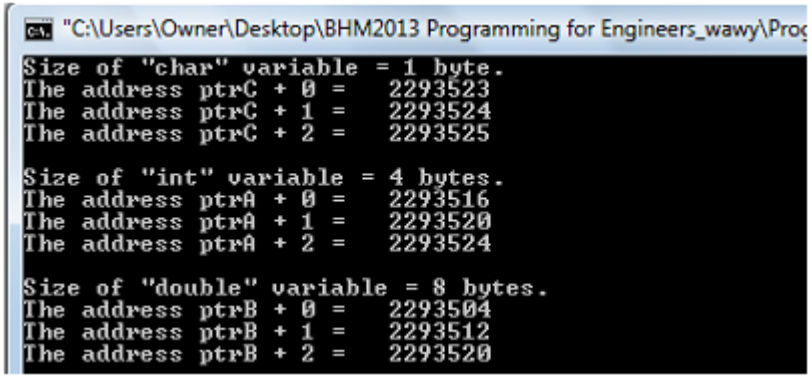
```
int* ptrA = &a;
ptrA = ptrA + 1;
```

2.0 Pointer Arithmetic

- Let's say the address of a (&a) is 2293535. What is now the value of ptrA+1? Is it 2293636, which is one address ahead? The answer lies in the pointer arithmetic. The example program on the right illustrates the arithmetic.

```c
#include <stdio.h>
/* This program is to illustrate the pointer arithmetic
Written by: WAWY February 2012 */
int main()
{
    /*  Create the standard integer data type. */
    char c; int a; double b;
    /*  Now, let's declare and initialize pointers to the variables. */
    char* ptrC = &c; int* ptrA = &a; double* ptrB = &b;
    /*  Let's display the pointer increment */
    printf("Size of \"char\" variable = %d byte.\n",sizeof(c));
    printf("The address ptrC + 0 =\t %u\n",ptrC+0);
    printf("The address ptrC + 1 =\t %u\n",ptrC+1);
    printf("The address ptrC + 2 =\t %u\n",ptrC+2);
    printf("\n");
    printf("Size of \"int\" variable = %d bytes.\n",sizeof(a));
    printf("The address ptrA + 0 =\t %u\n",ptrA+0);
    printf("The address ptrA + 1 =\t %u\n",ptrA+1);
    printf("The address ptrA + 2 =\t %u\n",ptrA+2);
    printf("\n");
    printf("Size of \"double\" variable = %d bytes.\n",sizeof(b));
    printf("The address ptrB + 0 =\t %u\n",ptrB+0);
    printf("The address ptrB + 1 =\t %u\n",ptrB+1);
    printf("The address ptrB + 2 =\t %u\n",ptrB+2);
    /*  End the program. */
    return 0;
}
```

2.0 Pointer Arithmetic



- Notice that address moves one-by-one based on the size of the variable. A “char” variable is only one byte. So, the pointer to a “char” variable (declared as char*) is a pointer to one byte information. On the other hand, since the “int” variable is a 4-byte data type, the pointer to an “int” variable (declared as “int*) is a 4-byte information. For the case of “double” variable, the size is 8-byte. Therefore, the pointer to double data type moves one-by-one based on 8-byte size

3.0 How to declare and use pointers?

- We can also make collection of similar structure data type by using an array of structure. We show an example of such array below.

```
int* ptrA;
double* ptrB;
```

- The * operator can be adjacent to the type declaration or to the identifier or can stand alone as long as it is in the middle between the type declaration and the identifier. All declarations stated below are acceptable.

```
int*  ptrA;  int  *  ptrA;  int  *ptrA;
```

3.0 How to declare and use pointers?

- To initialize the value of the pointer variable, we should assign to the address of existing variable, to other pointer or to the NULL pointer.

```
intA;                  //create integer variable named
A int* ptrA;           //create integer pointer variable named ptrA
int* ptrB;             //create integer pointer variable named
ptrB ptrA = &A;        //assign ptrA to the address of variable A
ptrB = ptrA;           //assign ptrB to the value of
ptrA ptrA = NULL;      //assign ptrA to NULL pointer
```

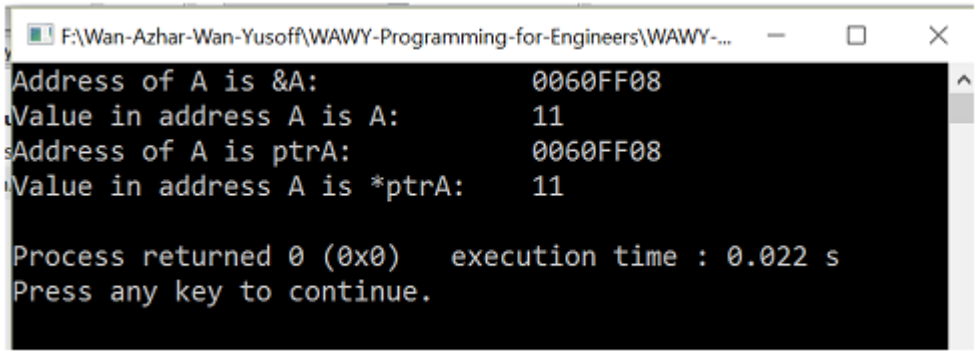
- The * operator can be adjacent to the type declaration or to the identifier or can stand alone as long as it is in the middle between the type declaration and the identifier. All declarations stated below are acceptable.
- The NULL pointer is a pointer to zero address. We use NULL pointer in order to make sure that every pointer has address to point to, instead of letting the pointer without address.

3.0 How to declare and use pointers?

- What is the important of knowing the address? When we have an address, we can get the value of what is in the address. To do that, we will use indirection operator which is also a *. The example is given below.

```
#include <stdio.h>
#include <stdlib.h>
 int main()
 {
     int A = 11;
     int *ptrA = &A;
     printf("Address of A is &A:\t\t%p\n",&A);
     printf("Value in address A is A:\t%d\n",A); //indirection
     printf("Address of A is ptrA:\t\t%p\n",ptrA);
     printf("Value in address A is *ptrA:\t%d\n",*ptrA); //indirection
     return 0;
}
```

3.0 How to declare and use pointers?



- In the above example, notice that the indirection operator is just the opposite of the address operator. Notice also that we use %p to print the address value. The format specifier of %x will give us hexadecimal representation.
- The notation *, at least has three usages: (1) to declare or represent pointer data type like int * (2) to perform indirection i.e. to get value pointed by the address like *b (3) and multiplication like a * b. We should be careful with the notation and always observe the actual meaning of the program.

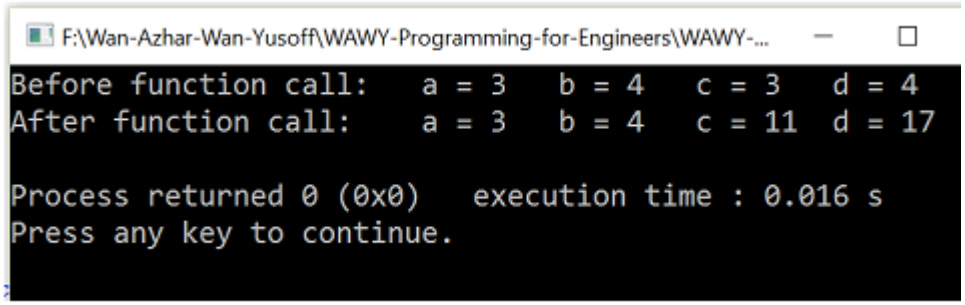
4.0 Pointers and function

- We can use pointers to solve the problem that every function must return only one value. We solve this by passing several pointer variables as parameters to function and let the function change the values through the given address. We call this passing parameters by reference (address) and not passing by value. We provide an example through the following program.

4.0 Pointers and function

```c
#include <stdlib.h>
int main()
{
    int a = 3;
    int b = 4;
    int c = 3;
    int d = 4;
    int *ptrC = &c;
    int *ptrD = &d;     //Before function
    printf("Before function call:\ta = %d\tb = %d\tc = %d\td = %d\n",a,b,c,d);
    passValue(a,b);
    passReference(ptrC,ptrD);     //After function
    printf("After function call:\ta = %d\tb = %d\tc = %d\td = %d\n",a,b,c,d);
    return 0;
}
void passValue(int a, int b)
{
    a = 11;
    b = 17;
    return;
}
void passReference(int *ptrC, int *ptrD)
{
    //indirection change the value
    *ptrC = 11;
    *ptrD = 17;
    return;
}
```

4.0 Pointers and function

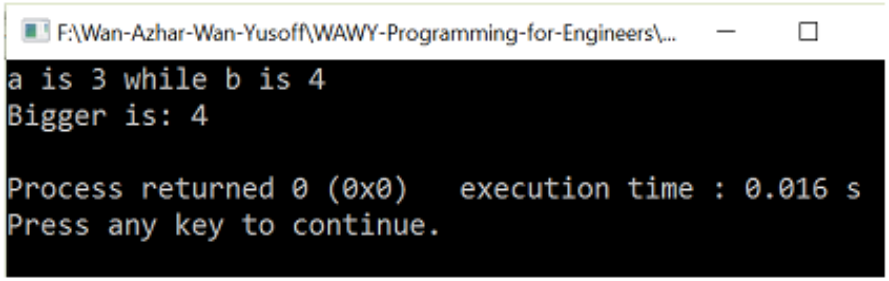


- In the above program, we pass the variable ‘a’ and ‘b’ the values while we pass the variable ‘c’ and ‘d’ as reference (address). Thus, when we change the value of ‘c’ and ‘d’ in the function, the value of ‘c’ and ‘d’ change. At the same time, we try to change the value of ‘a’ and ‘b’ but the values are unchanged. Using pointers, we can process many variables in a function and we don’t have to rely on the return value.
- We can also make a function to return a pointer. We give you an example below.

4.0 Pointers and function

```c
#include <stdio.h>
#include <stdlib.h>
int* checkValue(int *ptrA, int *ptrB);
int main()
{
    int a = 3;
    int b = 4;
    int *ptrA = &a;
    int *ptrB = &b;
    int* ptrCheck = NULL;
    ptrCheck = checkValue(ptrA,ptrB);
    printf("a is %d while b is %d\n",a,b);
    printf("Bigger is: %d\n",*ptrCheck);
    return 0;
}
int* checkValue(int *ptrA, int *ptrB)
{
    if (*ptrA > *ptrB)  //check content a and b
        return ptrA;
    else
        return ptrB;    //return address a
}
```

4.0 Pointers and function



- Next time, we will look at pointer with array, string and memory allocation. Another reason we use pointer because most of the times, sharing address is more efficient than sharing data.