INTRODUCTION TO C POINTERS

Topics

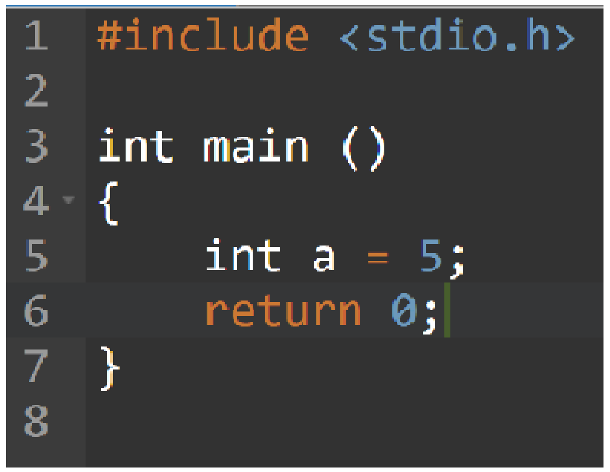
1. Introduction to C pointers
2. Declaration of pointers
3. Initialization of pointers
4. Getting and setting values using pointers
5. Incrementing/Decrementing a pointer
6. Pointer addition/subtraction

1. Introduction to C Pointers

- When program is executed, its variables are stored randomly in memory (RAM).

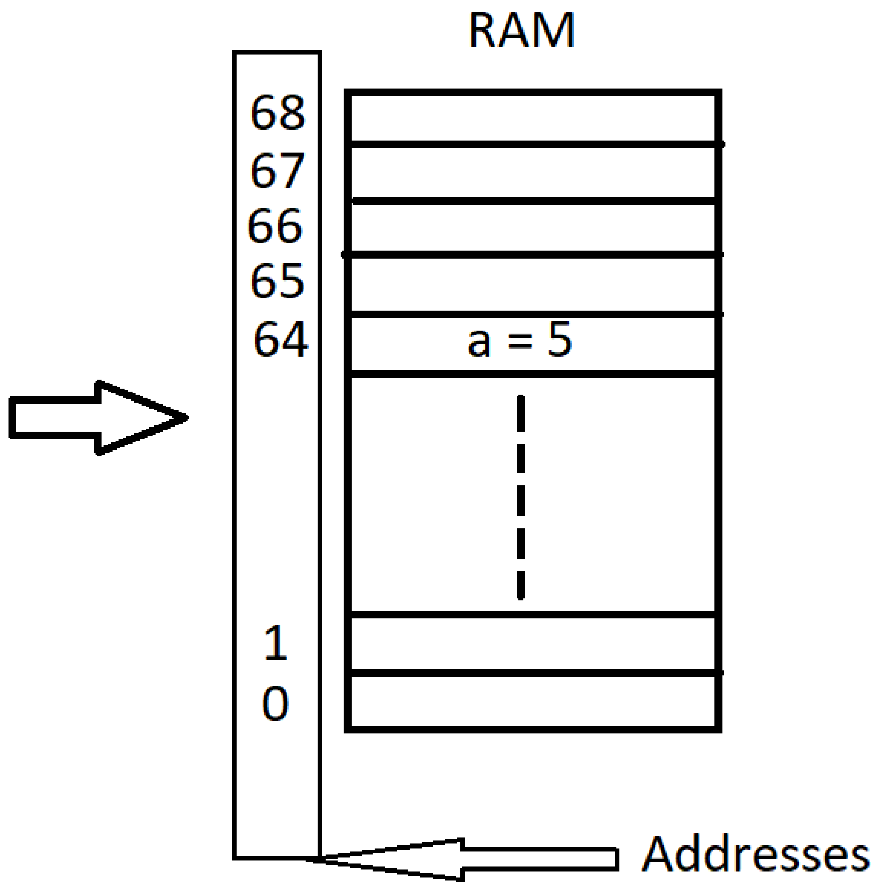
- Each variable has a value, and an address.

Storage of a variable in RAM



```
#include <stdio.h>

int main ()
{
	int a = 5;
	return 0;
}
```

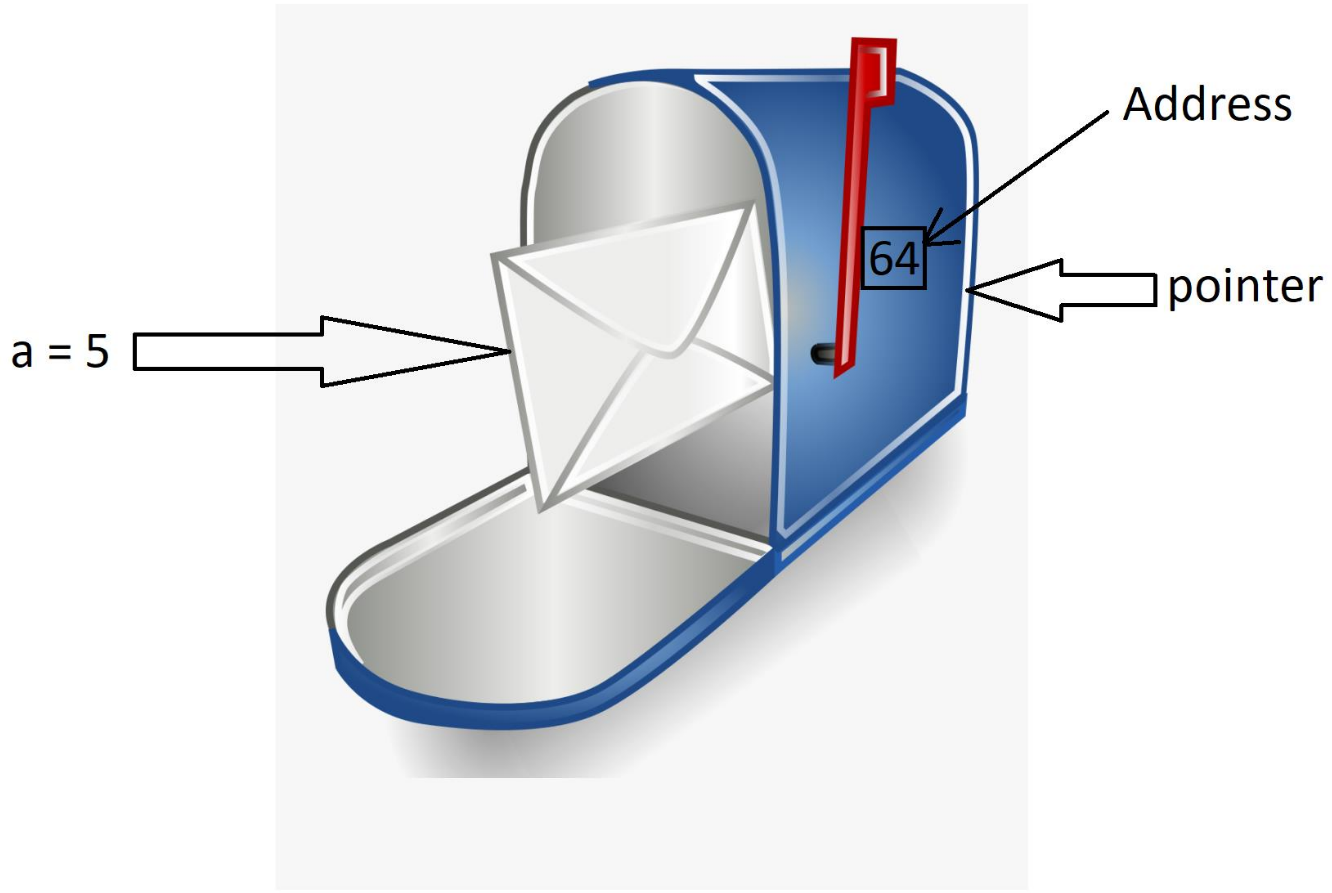


Definition of a pointer

- A pointer is a variable whose value is the address of another variable.

- Advantages of pointers:
  - Enhance execution speed of a program
  - Reduces the storage space and complexity of the program
  - Allows a function to return multiple values

Analogy of a pointer to a mailbox



Next: Declaration

2. Declaration of pointers

- A pointer can hold the address to a character (char), integer (int), float (float), and other data types or data structures.
- A pointer is therefore declared by stating its name, data type it is pointing to, and an asterisk
- Examples
  - `int* ptr1;        // A pointer to an integer`
  - `char * ptr2;      // A pointer to a character`
  - `float *ptr3;      // A pointer to a float`

Generic (Void) pointer

- A special type of pointer, called “generic pointer”, is declared with the “void” keyword.
- This pointer is not associated with any data type, therefore address of any data type can be assigned to it.
- Void pointers are useful when data type of target variable is not known before hand.

Example

```
void * ptr1;            // Declaration of a general purpose pointer
```

Next: Initialization

3. Initialization of pointers

- Once a pointer is declared, its value must be assigned to the address it points to (or NULL is address is not known beforehand)
- This is done using the ampersand (&) symbol in front of the name of another variable

```
int a = 5;              // Declare an integer whose value is 5
int * ptr1;             // Declare a pointer to an integer
void * ptr2;            // Declare a generic pointer
int * ptr3;             // Declare another pointer to an integer
ptr1 = &a;              // The value of ptr1 is now the address of a
ptr2 = &a;              // The value of ptr2 is also the address of a
ptr3 = NULL;            // The value of ptr3 is 0
```

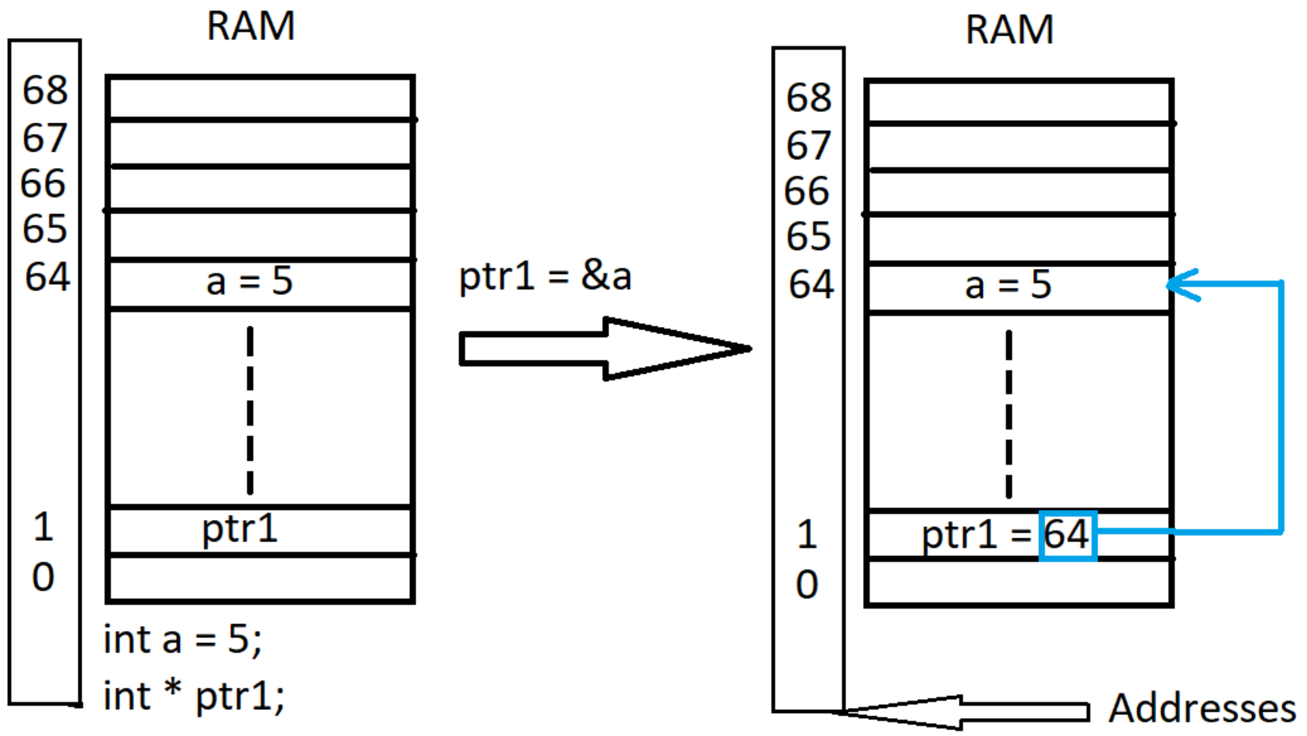
C Code: https://onlinegdb.com/rkkydo2Or

QUESTION

Q1. Where are pointers stored?

Answer: https://onlinegdb.com/S1OtBjhur

Summary: Declaration and initialization of a pointer



```
int * ptr1 = &a; // Simulataneous declaration and initialization of a pointer
```

4.1. Getting values using pointers

- A pointer can “get” or access the value of a variable via the address of that variable
- This is done by prepending the asterisk (*) operator to the name of the pointer.

Example

```
int a = 5;
int * ptr1;
ptr1 = &a;
printf("%d",*ptr1);        // Prints 5, which is the value of a.
```

Therefore, a and *ptr1 are the same (5)

Asterisk in Declaration and Accessing values

- During declaration, the asterisk (*) indicates that the declared variable is a pointer.

  Therefore:

  int * ptr1;   //declaration of a pointer called ptr1 that points to an integer

- After declaration, the asterisk (*) is used to “get” the contents pointed by ptr1. This is referred to as “dereferencing” or “indirection”.

  Thus:

  *ptr1 means “content at (address) ptr1”. (i.e. *ptr1 = a = 5)

C Code: https://onlinegdb.com/BkPiNo3ur

Getting values via Void pointers [1/2]

- Since the type of a void pointer is not declared beforehand, special care my be taken when dereferencing.

For example, the following code gives an error when executed.

```
int a = 5;              // Declare an integer whose value is 5
void* ptr1;             // Declare a void pointer
ptr1 = &a;              // The value of the void pointer is now the address of a
printf(“%d”,*ptr1);     // Gives an error
```

C Code: https://onlinegdb.com/SkF8KjnuS

- The error happens because the compiler does not have any clue about the type of the value pointed to.

Getting values via Void pointers [2/2]

- To overcome the error, the void pointer must be “casted” to the type of the variable it points to. For example, casting a void pointer to an integer is done by:

```
(int *)ptr1;      // Casts void pointer to integer
```

- The value of (int*)ptr1 is now the “address” of the integer a = 5

- Thus, the “value” of a is given by *((int*)ptr1), and the following print statement will not give errors

```
printf(“%d”,*((int*)ptr1));      // NO ERROR
```

C Code : https://onlinegdb.com/ryBVM23OS

4.2 Setting values using pointers

- A pointer can modify, or mutate, the value of a variable via the address of the variable. For example:

```
int a = 5;
int * ptr1;
void * ptr2;
ptr1 = &a;
Ptr2 = &a;
*ptr1 = 20;                    // Set the value of a to 20 via ptr1.
printf(“%d”,a);                // Prints 20
*((int*)ptr2) = 40; // Set the value of a to 40 via ptr2
printf(“%d”,a);                // Prints 40
```

C Code: https://onlinegdb.com/H1XRD32Or

QUESTION

Given the following:

```
int a = 5;
int * ptr1;
ptr1 = &a;
ptr1 = 20;
```

Q2. What would happen when the following print statements are called?

```
printf(“%d”,a);
Printf(“%d”,*ptr1);
```

Answer: https://onlinegdb.com/S1aXLn2dH

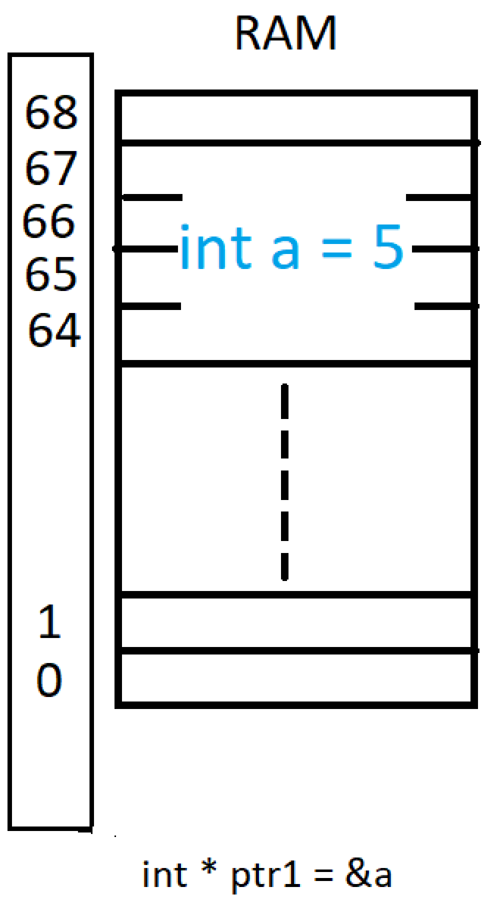
5. Incrementing/Decrementing a pointer

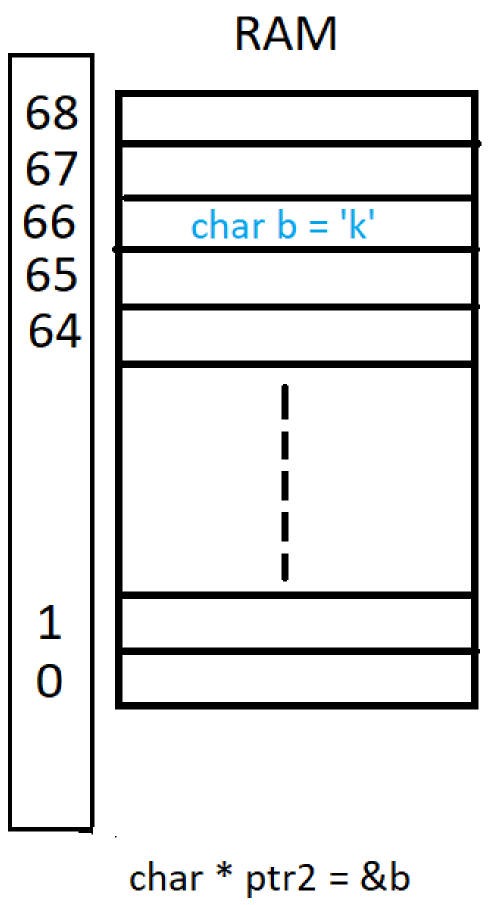
- The memory (RAM) is organized in units called bytes.
- Each data type occupies one or more bytes inside the memory.

  For example:
  - Integer (int) requires 4 bytes of memory.
  - Character (char) requires 1 byte of memory.
  - Float (float) requires 4 bytes of memory.

Storage of Data Types in Memory

- The starting address of a is 64
- The starting address of b is 66
- ptr1 = 64;
- ptr2 = 66;





Incrementation/Decrementation Example

When incrementing/decrementing a pointer by 1, its value(address it points to) increases/decreases by the size of the datatype it points to.

For example

int a = 5;   char b = ‘k’;

int * ptr1 = &a;   char * ptr2 = &b;

ptr1+1 = 64 + sizeof(integer) = 64 + 4 = 68

Ptr2+1 = 66 + sizeof(character) = 66 +1 = 67

Ptr1-1 = 64 - sizeof(integer) = 64 - 4 = 60

Ptr2-1 = 66 - sizeof(character) = 66 -1 = 65

C Code : https://onlinegdb.com/r1k8P22_r

6. Pointer Addition/Subtraction

- Pointer addition and subtraction is the general case of incrementation/decrementation of a pointer by integer multiple of the size of the datatype it points to.

Example

int a = 5;

int * ptr1 =&a;

ptr1 + 3 = 64 + 3*[sizeof(integer)] = 64 + 3*4 = 76

ptr2 – 5 = 66 – 5 [sizeof(character)] = 66 – 5*1 = 61

CHEAT SHEET AND EXERCISES

- Cheat sheet on C pointers

- EXERCISES
  - Program to create, initialize, assign and access a pointer variable.
  - Program to print size of different types of pointer variables.
  - An Example of Null pointer in C
  - Modify value stored in other variable using pointer in C

REFERENCES

- Brian W. Kernighan. 1988. The C Programming Language (2nd ed.). Prentice Hall Professional Technical Reference.
- Zhang, Tony. Sams teach yourself C in 24 hours. Indianapolis, Ind: Sams, 2000. Print

THE END

QUESTIONS