

This lesson will discuss further on the concept of Pointer Variable and its application.

OVERVIEW

The main aspects of Pointers are as follows:

- Address Operator
- **Pointer Variables**
- Call by Reference

Content Copyright Nanyang Technological University

2

We recalled that there are 3 main aspects of Pointers to be learnt.
In this lesson, we will focus on Pointer Variables.

POINTER VARIABLES

Lesson on Pointer Variables are as follows:

- 5.2.1: Pointer Variables (Part 1 of 4)
- 5.2.2: Pointer Variables (Part 2 of 4)
- 5.2.3: Pointer Variables (Part 3 of 4)
- 5.2.4: Pointer Variables (Part 4 of 4)

Content Copyright Nanyang Technological University

3

This lesson on Pointer Variables will be covered in 4 parts: 5.2.1, 5.2.2, 5.2.3 and 5.2.4.

This lesson is 5.2.1 and it covers all the main concepts of pointer variables.

Lesson 5.2.2 and 5.2.3 are detailed examples of how to apply pointer variables in programming codes.

Lesson 5.2.4 will be a summary to the lesson on Pointer Variables.

LEARNING OBJECTIVES

By the end of this lesson, you should be able to:

Content Copyright Nanyang Technological University

4

By the end of this lesson, you should be able to:

LEARNING OBJECTIVES

By the end of this lesson, you should be able to:

- Explain the use of pointer variables

Explain the use of pointer variables

LEARNING OBJECTIVES

By the end of this lesson, you should be able to:

- Explain the use of pointer variables
- Apply pointer variables

Apply pointer variable

LEARNING OBJECTIVES

By the end of this lesson, you should be able to:

- Explain the use of pointer variables
- Apply pointer variables
- Apply indirection operator

Apply indirection operator

LEARNING OBJECTIVES

By the end of this lesson, you should be able to:

- Explain the use of pointer variables
- Apply pointer variables
- Apply indirection operator
- Apply pointer variables by assigning variable address to pointer variables

Apply pointer variables by assigning variable address to pointer variables

POINTER VARIABLES: DECLARATION

Pointer variable

Content Copyright Nanyang Technological University

9

Pointer variable

POINTER VARIABLES: DECLARATION

Pointer variable

- Different from the variable of primitive data type such as int, float, char

Content Copyright Nanyang Technological University

10

Pointer variable is different from the variables of primitive data types which stores the values directly

POINTER VARIABLES: DECLARATION

Pointer variable

- Different from the variable of primitive data type such as int, float, char
- Stores the **address** of memory location of a data object

Content Copyright Nanyang Technological University

11

A pointer variable stores the addresses of memory locations of a data object

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

Content Copyright Nanyang Technological University

12

A *pointer variable* is declared by:

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

data_type

Content Copyright Nanyang Technological University

13

data_type

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

`data_type * ptr_name`

Content Copyright Nanyang Technological University

14

`asterisk pointer_name;`

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

data_type * ptr_name

Content Copyright Nanyang Technological University

15

where **data_type** can be any C data type such as

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

char ————— **data_type** * **ptr_name**

Content Copyright Nanyang Technological University

16

Character

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

```
char ————— data_type * ptr_name  
int
```

Content Copyright Nanyang Technological University

17

Integer

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

```
char   data_type * ptr_name  
int  
float
```

Content Copyright Nanyang Technological University

18

Float

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

```
char   data_type * ptr_name  
int  
float  
double
```

Content Copyright Nanyang Technological University

19

or **double**

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

```
char   data_type * ptr_name  
int  
float  
double
```

Content Copyright Nanyang Technological University

20

The **data_type** is used to indicate the type of data that the variable is pointed to.

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,

```
char   data_type * ptr_name  
int  
float  
double
```

Content Copyright Nanyang Technological University

21

Pointer_name is the name of the pointer variable.

POINTER VARIABLES: DECLARATION

A **pointer variable** is declared by,



Content Copyright Nanyang Technological University

22

An asterisk (*) is used to indicate that the variable is a pointer variable.

POINTER VARIABLES: DECLARATION

An example of declaring
pointer variable:

```
char   data_type * ptr_name
      +-----+
      |           |
int    float   double
```

Content Copyright Nanyang Technological University

23

Let's take a look at an example of declaring pointer variable.

POINTER VARIABLES: DECLARATION

An example of declaring
pointer variable:

```
int *ptrI;
```

char data_type * ptr_name
int float double

Content Copyright Nanyang Technological University

24

If we want to name the pointer variable as pointer I, we can use the short form p t r I to represent.

POINTER VARIABLES: DECLARATION

An example of declaring
pointer variable:

```
int *ptrI;
```



The diagram illustrates the components of a pointer declaration. It shows a central box labeled "data_type" with an asterisk (*) and a pointer name "ptr_name" attached. To the left of this box are four data types: "char", "int", "float", and "double". Lines connect each of these four types to the "data_type" box, indicating they are all valid types for a pointer variable.

Content Copyright Nanyang Technological University

25

So the declaration statement would be
integer asterisk pointer i

Which can be also be written in the following ways as well.

POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

int *ptrI;
or int * ptrI;

char data_type * ptr_name
int float double

Content Copyright Nanyang Technological University

26

[pause for 1 sec]

POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

```
int *ptrI;  
or    int * ptrI;  
or    int* ptrI;
```

char data_type * ptr_name
int |
float |
double |

Content Copyright Nanyang Technological University

27

[pause for 1 sec]

POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

```
int *ptrI;  
or int * ptrI;  
or int*ptrI;
```



ptrI

Content Copyright Nanyang Technological University

28

All these 3 statements are the same. The statement declare a pointer variable with the use of the asterisk sign.

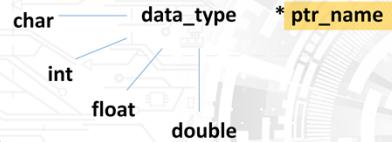
Notice that the whitespace in between the asterisk does not matter.

POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

```
int *ptrI;  
or    int * ptrI;  
or    int* ptrI;
```



ptrI

Content Copyright Nanyang Technological University

29

The pointer variable declared is named as pointer I in this example.

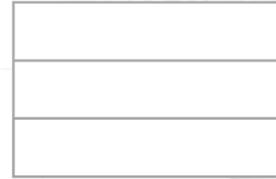
POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

```
int *ptrI;  
or    int * ptrI;  
or    int*ptrI;
```

ptrI



Content Copyright Nanyang Technological University

30

The pointer variable points to the address of a memory location.
The C P U has many memory spaces represented by the boxes as shown.

POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

```
int *ptrI;  
or int * ptrI;  
or int*ptrI;
```



ptrI

Memory address 102C

Memory address 1028

Memory address 1024



Content Copyright Nanyang Technological University

31

Each of these memory is unique and can be identified by memory address such as 1024, 1028, 102C etc.

POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

```
int *ptrI;  
or int * ptrI;  
or int*ptrI;
```



Memory address 102C

Memory address 1028

Memory address 1024

Integer Value Stored
(4 bytes)

ptrI

Content Copyright Nanyang Technological University

32

For this example, let's say the CPU has allocated memory address 1024 to store an integer value.

And the job for pointer I is to point to this memory location.

POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

```
int *ptrI;  
or int * ptrI;  
or int*ptrI;
```



Memory address 102C

Memory address 1028

ptrI 1024

Memory address 1024

Integer Value Stored
(4 bytes)

Content Copyright Nanyang Technological University

33

So the pointer I has the value 1024 stored in it.

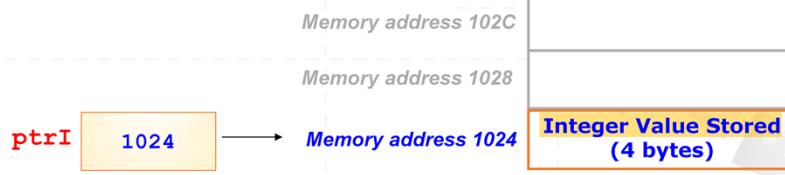
It means that pointer I will point to the memory address of 1024 as shown in the diagram.

POINTER VARIABLES: DECLARATION

An example of declaring

pointer variable:

int *ptrI;
or int * ptrI;
or int* ptrI;



Content Copyright Nanyang Technological University

34

At the address that was pointed to, it is used to store an **integer**. That is why the data type of the pointer I is integer

POINTER VARIABLES: DECLARATION

An example of declaring
pointer variable:

```
int *ptrI;  
or    int * ptrI;  
or    int* ptrI;
```

ptrI is a pointer variable.



Content Copyright Nanyang Technological University

35

Note that pointer I is a pointer variable

POINTER VARIABLES: DECLARATION

An example of declaring
pointer variable:

```
int *ptrI;  
or    int * ptrI;  
or    int* ptrI;
```

ptrI is a pointer variable. It stores the address of the memory which is used for storing an Int value.



Content Copyright Nanyang Technological University

36

The value of a pointer variable is an **address**, which is different from other variables of primitive data types that store the **data** directly.

In this example, it stores the **address** of the memory which is used for storing an integer value.

POINTER VARIABLES: DECLARATION

An example of declaring
pointer variable:

```
int *ptrI;  
or    int * ptrI;  
or    int* ptrI;
```

ptrI is a pointer variable. It stores the address of the memory which is used for storing an Int value. It does **not** store the **value** of the variable directly.



Content Copyright Nanyang Technological University

37

It does **not** store the **value** of the variable. It stores the **address** of the memory.

POINTER VARIABLES: DECLARATION

An example of declaring
pointer variable:

```
int *ptrI;  
or    int * ptrI;  
or    int* ptrI;
```

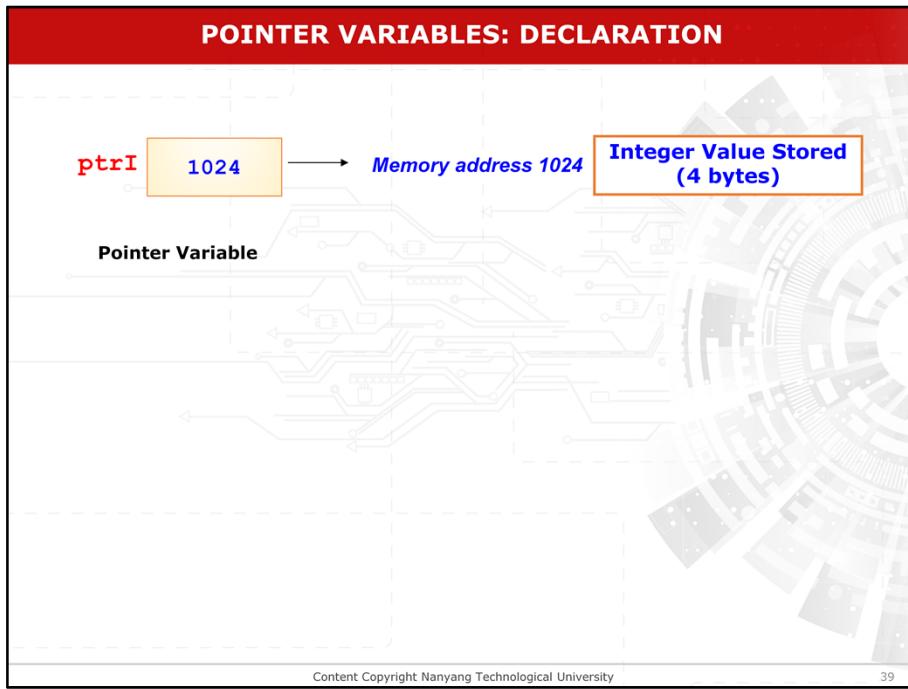
ptrI address
*ptrI actual value



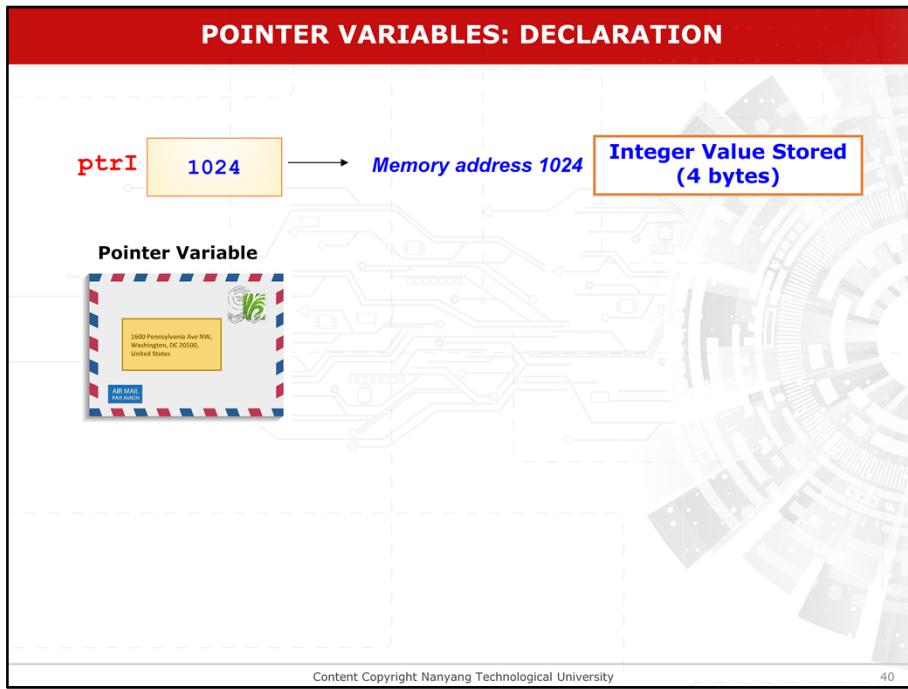
Content Copyright Nanyang Technological University

38

If we want to retrieve the actual value, we will need to use indirection operator (*),
e.g. ***ptr i**.



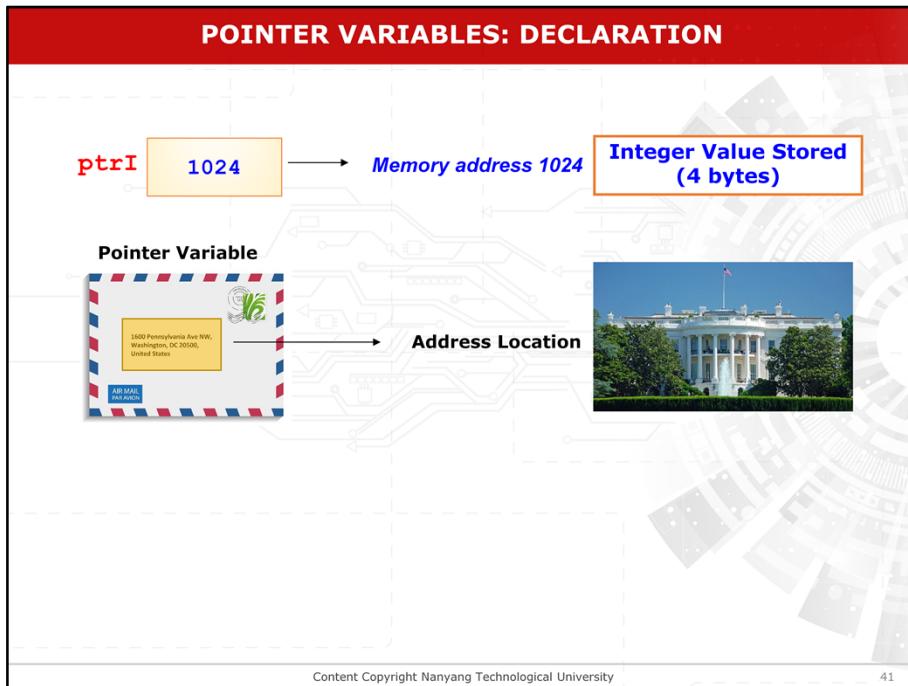
Pointer variable is similar to



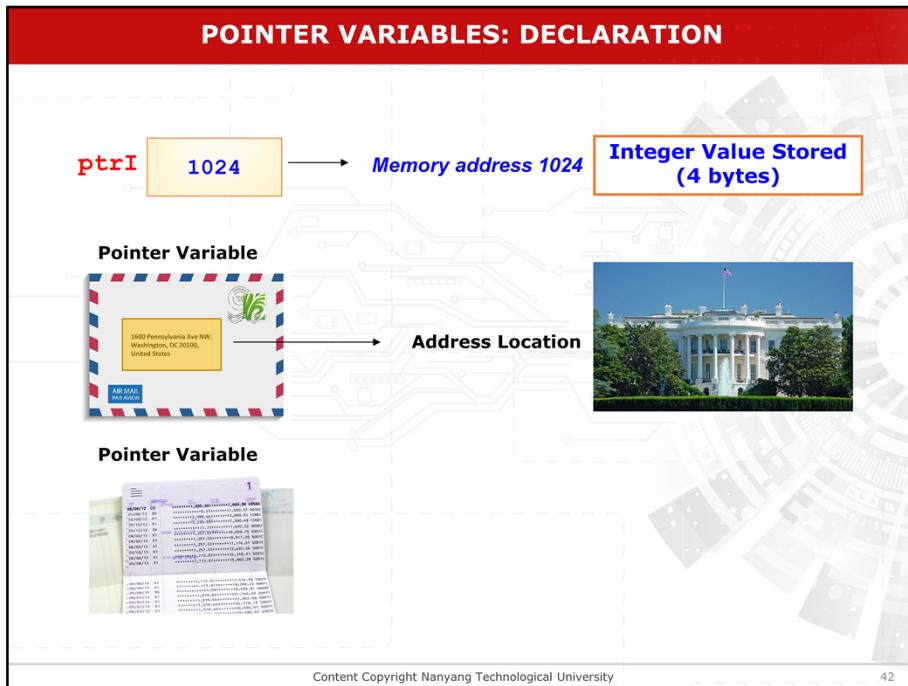
Content Copyright Nanyang Technological University

40

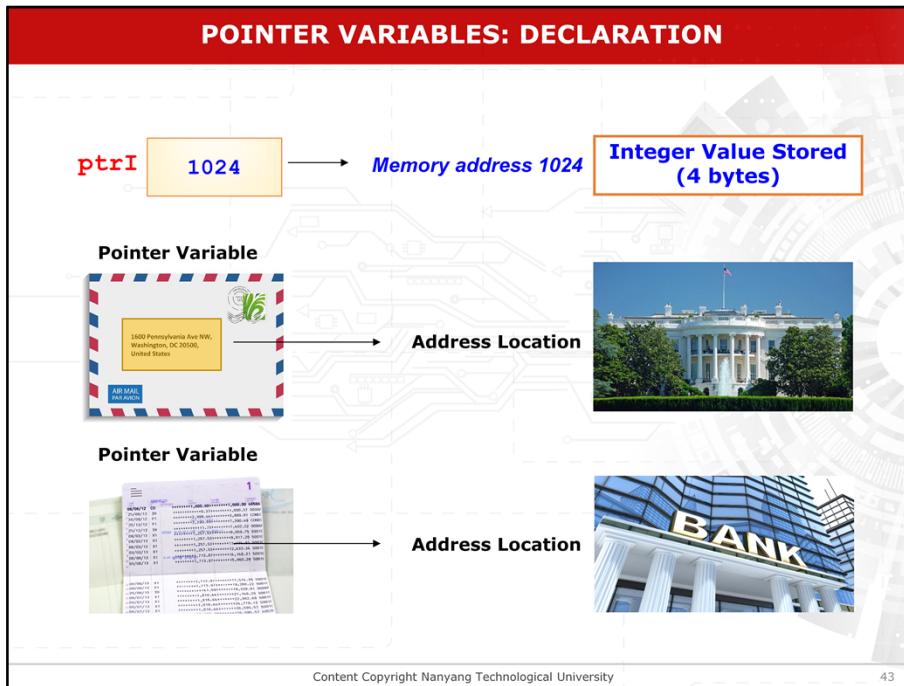
the address written on an envelope which stores the home address



and the actual place can be referred to by the address



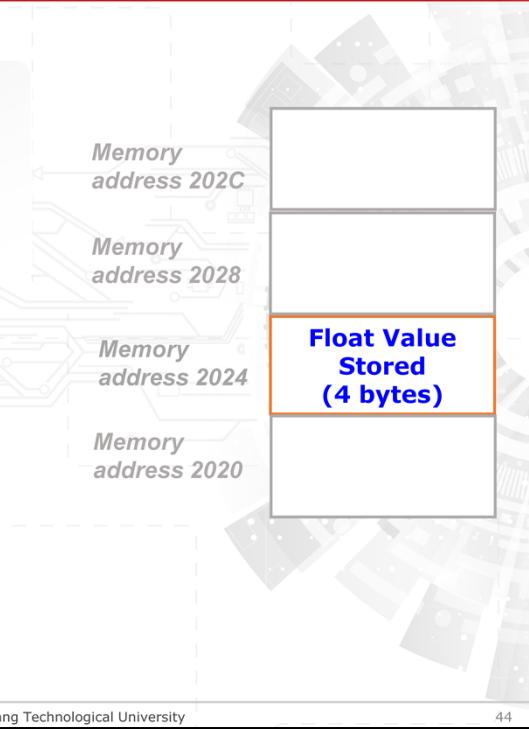
Pointer variable is also similar to a bank account, which contains the saving information



and the bank location that stores the real saving money. The saving money can be referred to via the bank account.

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing float value:



Content Copyright Nanyang Technological University

44

Let's see an example of declaring a **pointer variable** for storing float value.

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing float value:

```
float *ptrF;
```

Content Copyright Nanyang Technological University

If we want to name the pointer variable as pointer F which we will use the short form p t r F, the statement of declaration would be

Float asterisk pointer F

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing float value:

```
float *ptrF;
```

ptrF

Float Value
Stored
(4 bytes)

ptrF is a pointer variable.

Content Copyright Nanyang Technological University

46

Pointer F is a pointer variable.

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing float value:

```
float *ptrF;
```

ptrF

2024

Memory address 202C

Memory address 2028

Memory address 2020

Memory address 2024

**Float Value
Stored
(4 bytes)**

ptrF is a pointer variable. It can be used to store the **address** of the memory (e.g. 2024 in this case)

Content Copyright Nanyang Technological University

47

It can be used to store the address of the memory (for example 2024 in this case)

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing float value:

```
float *ptrF;
```

ptrF

2024

Memory address 202C

Memory address 2028

Memory address 2024
Float Value Stored (4 bytes)

Memory address 2020

ptrF is a pointer variable. It can be used to store the **address** of the memory (e.g. 2024 in this case) that is used to store a float value.

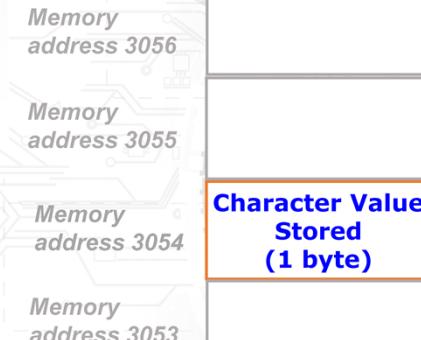
Content Copyright Nanyang Technological University

48

that is used to store a **float**.

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing character value:



Content Copyright Nanyang Technological University

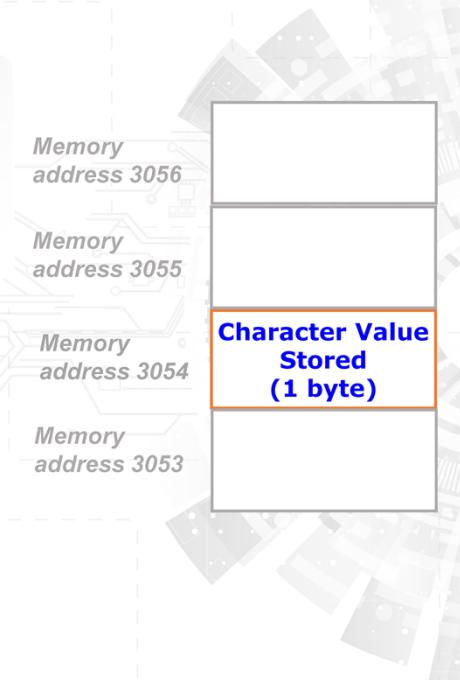
49

Let's see an example of declaring a **pointer variable** for storing character value.

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing character value:

```
char *ptrC;
```



Content Copyright Nanyang Technological University

50

If we want to name the pointer variable as pointer C which we will use the short form p t r C, the statement of declaration would be

Character asterisk pointer C

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing character value:

```
char *ptrC;
```

ptrC

Character Value
Stored
(1 byte)

ptrC is a pointer variable.

Content Copyright Nanyang Technological University

51

Pointer C is a pointer variable.

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing character value:

```
char *ptrC;
```

ptrC

3054

Memory address 3056

Memory address 3055

Memory address 3053

Character Value
Stored
(1 byte)

ptrC is a pointer variable. It can be used to store the **address** of the memory (e.g. 3054 in this case)

Content Copyright Nanyang Technological University

52

It can be used to store the address of the memory (for example 3054 in this case)

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing character value:

```
char *ptrC;
```

ptrC 3054

Memory address 3056

Memory address 3055

Memory address 3054

Memory address 3053

Character Value
Stored
(1 byte)

ptrC is a pointer variable. It can be used to store the **address** of the memory (e.g. 3054 in this case) that is used to store a character

Content Copyright Nanyang Technological University

53

that is used to store a **character**.

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing character value:

```
char *ptrC;
```

ptrC

Memory address 3056

Memory address 3055

Memory address 3054

Memory address 3053

Content Copyright Nanyang Technological University

54

We need to note that in the statement Character asterisk pointer C, the pointer C is declared without initialization, that is no value has been assigned to it yet.

POINTER VARIABLES: DECLARATION

Example of declaring a **pointer variable** for storing character value:

```
char *ptrC;
```

ptrC

Memory address 3056

Memory address 3055

Memory address 3054

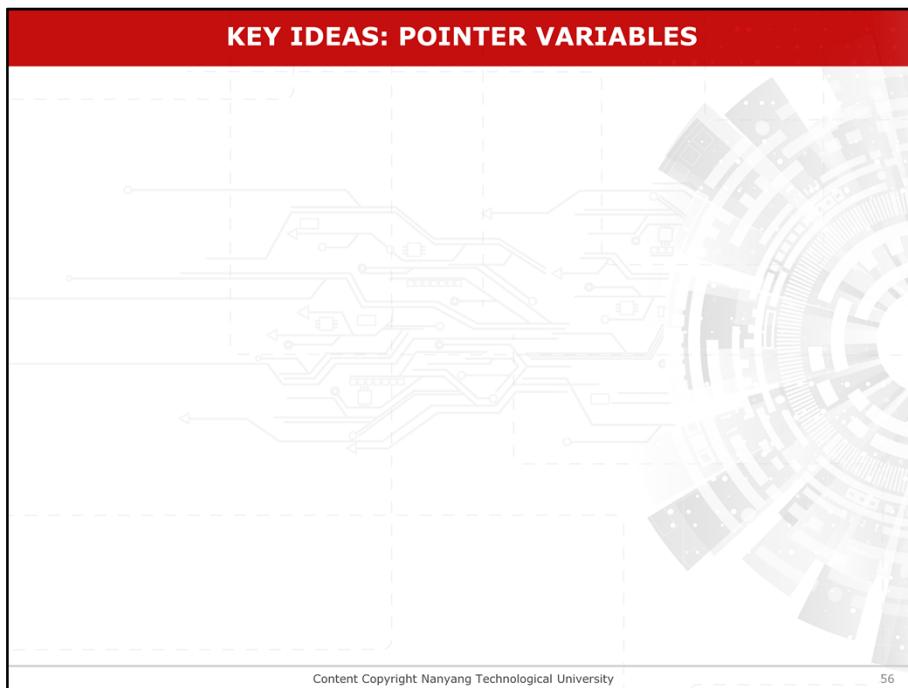
Memory address 3053

When a pointer is declared without initialization, memory is allocated to the pointer variable. However, no data or address is stored there.

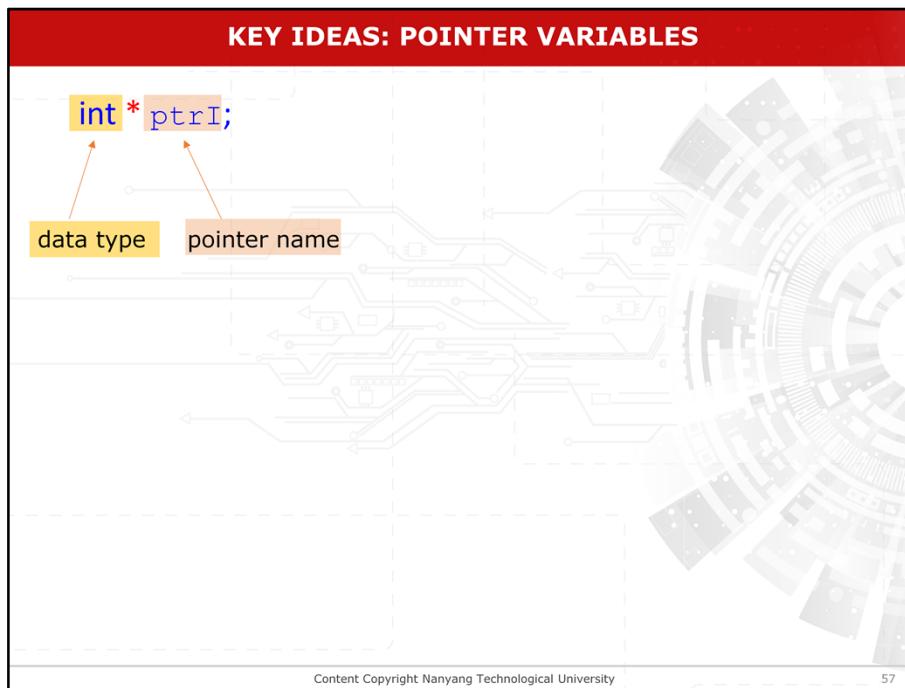
Content Copyright Nanyang Technological University

55

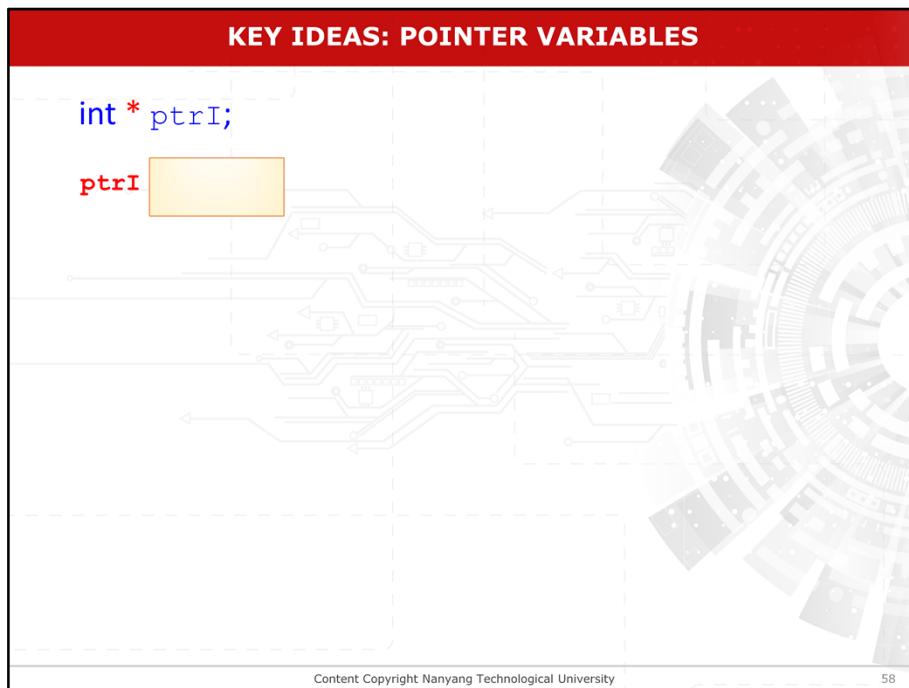
When a pointer is declared without initialization, memory is allocated to the pointer variable. However, no data or address is stored there yet.



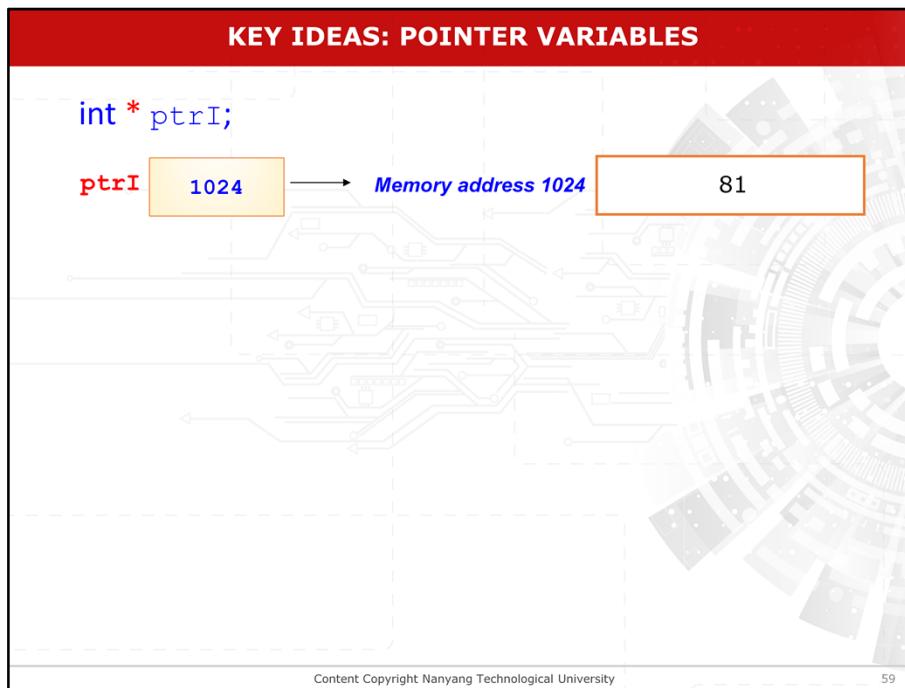
Let us recap the key ideas.



Pointer variable is declared with the format data type asterisk pointer name.



For example, integer asterisk pointer i. After this statement is declared, the CPU will allocate memory for pointer i.



Pointer variable stores an address which refers to the location that stores the actual data.

KEY IDEAS: POINTER VARIABLES

```
int * ptrI;
```

ptrI 1024 → Memory address 1024 81

Pointer variable: ptrI

Value of the variable is an
address

Above example: ptrI = 1024

Content Copyright Nanyang Technological University

60

So pointer I is a pointer variable and the value of the pointer variable is an address. In this example, pointer I equals 1024.

KEY IDEAS: POINTER VARIABLES

The diagram illustrates the concept of pointer variables and indirection. It shows a memory model where a pointer variable `ptrI` contains the memory address 1024. The value at memory address 1024 is 81, which is the value stored in the variable `i`. The pointer variable `ptrI` is labeled `ptrI` with the value `1024`. An arrow points from `ptrI` to the label `Memory address 1024`, which is enclosed in a box. To the right of this box is the value `81`, preceded by the indirection operator `*ptrI`.

Pointer variable: `ptrI`

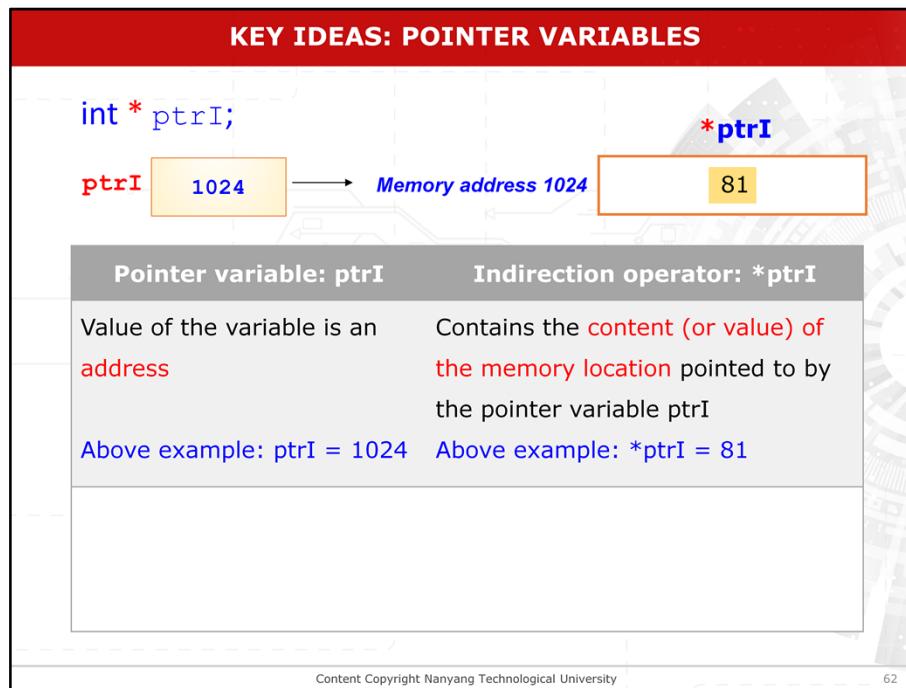
Indirection operator: `*ptrI`

Value of the variable is an address

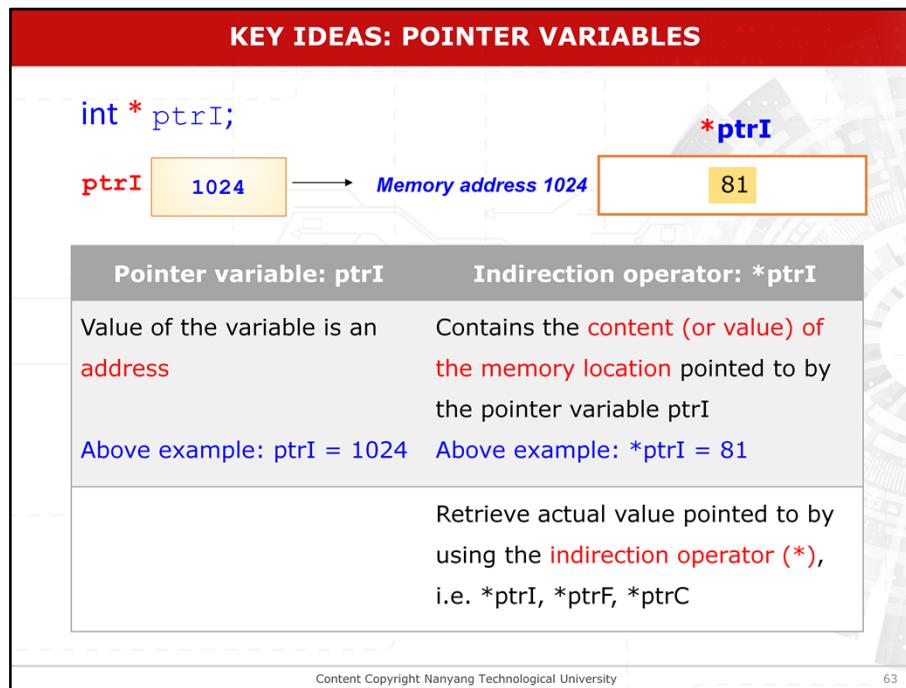
Above example: `ptrI = 1024`

Content Copyright Nanyang Technological University 61

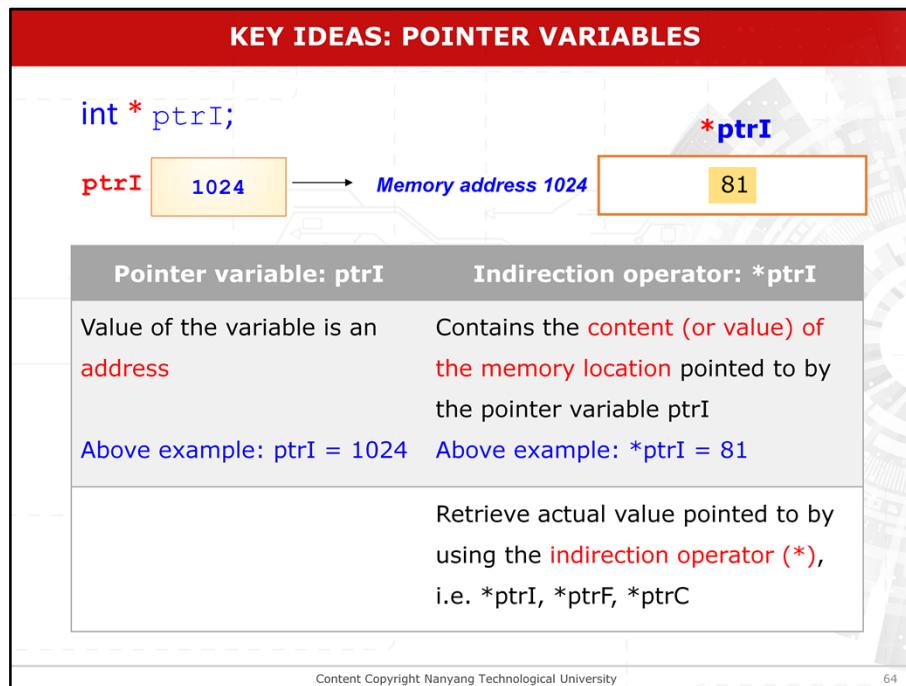
To retrieve the value stored in the pointed memory address, we use indirection operator, for example, asterisk pointer i.



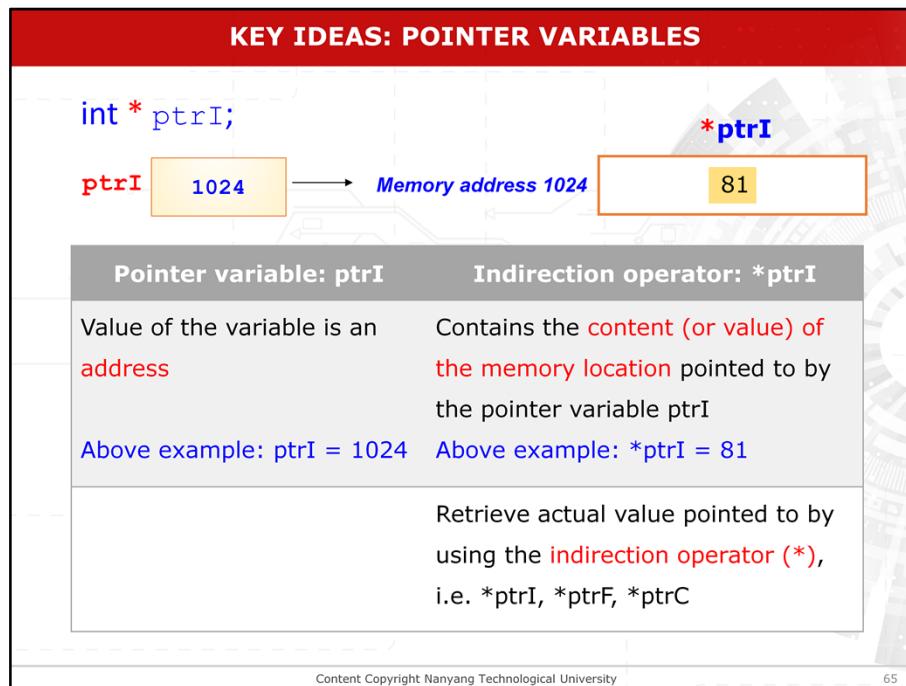
So asterisk pointer contains the **content (or value) of the memory location** pointed to by the pointer variable pointer i. In this case, the value of asterisk pointer I equals 81



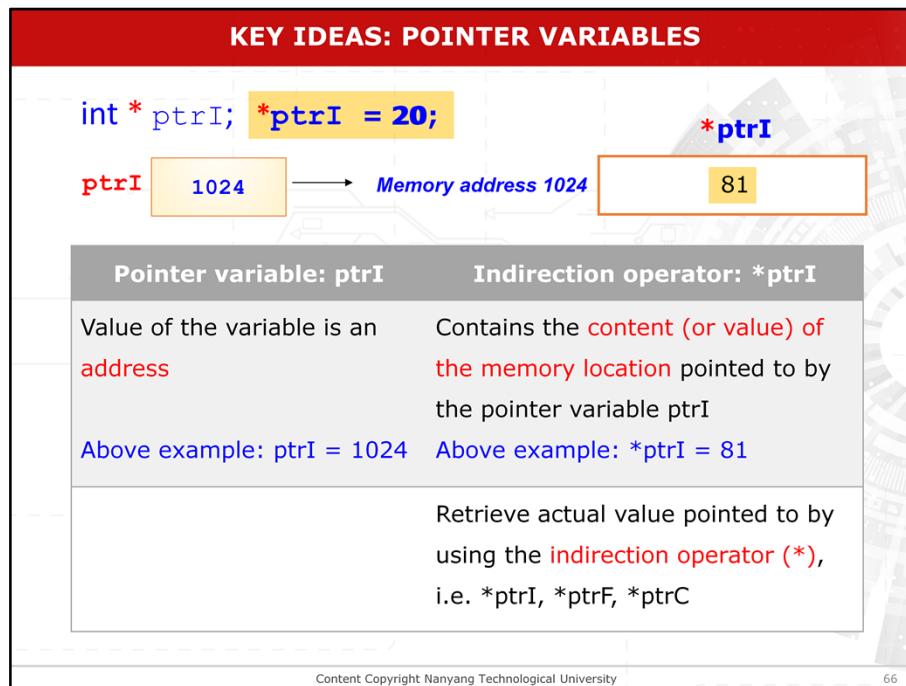
We can retrieve actual value pointed to by using the **indirection operator asterisk** that is asterisk pointer I, asterisk pointer F and asterisk pointer C and so on.



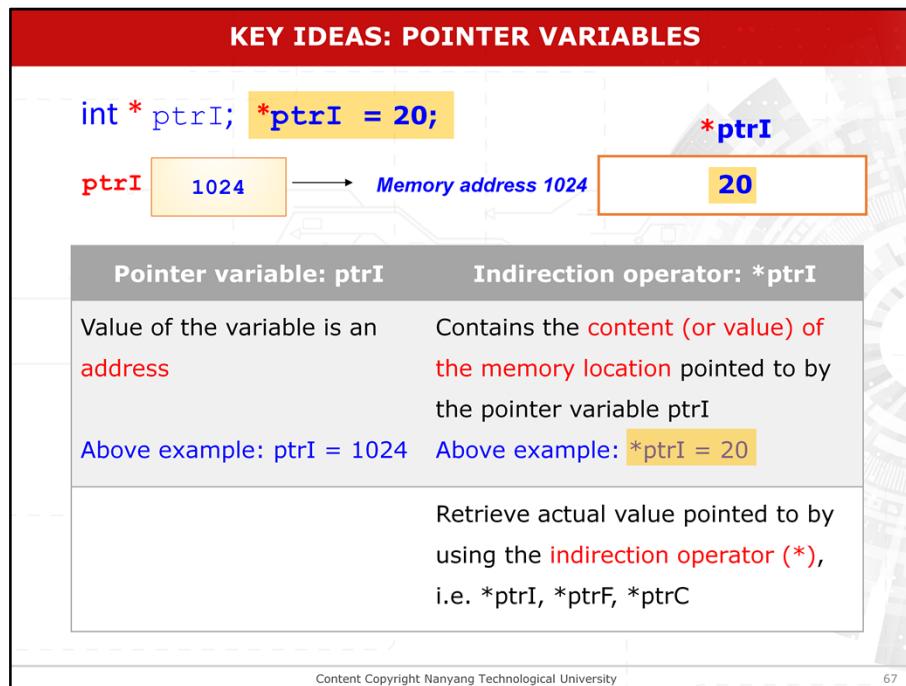
What if we want to change the value stored in the pointed memory?



We can make use of indirection operator as well.

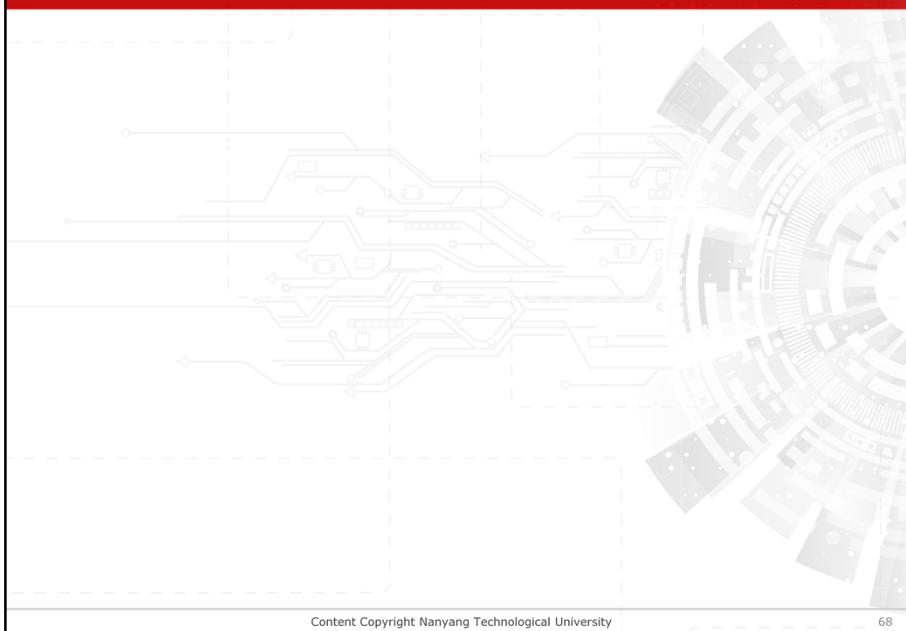


In this example, the statement asterisk pointer I equal 20 will update the value 81 to 20.



Therefore, asterisk pointer I equals 20 now.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES



Let's take a look at this example of Assigning Variable Address to Pointer Variable

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```



Content Copyright Nanyang Technological University

69

These 3 statements declared variables a, b, and c of primitive types integer, float, and character respectively.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

a

Addr: 1000

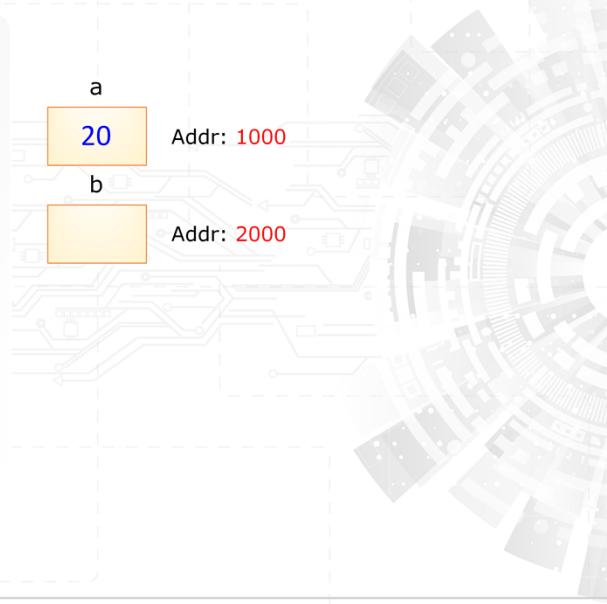
Content Copyright Nanyang Technological University

70

The CPU will allocate the memory address whenever variables are declared. For simplicity, let's assume that the CPU allocated memory address 1000 for variable a and store the integer value of 20.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```



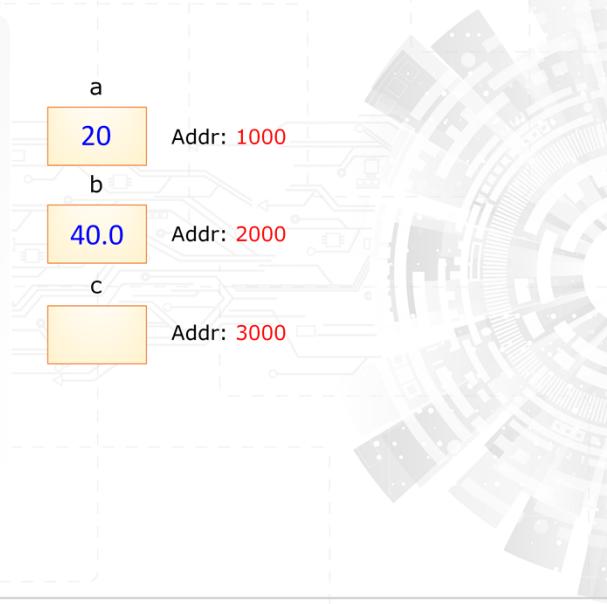
Content Copyright Nanyang Technological University

71

For variable `b`, the CPU allocated memory address 2000 for it to store a float value of 40.0.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```



Content Copyright Nanyang Technological University

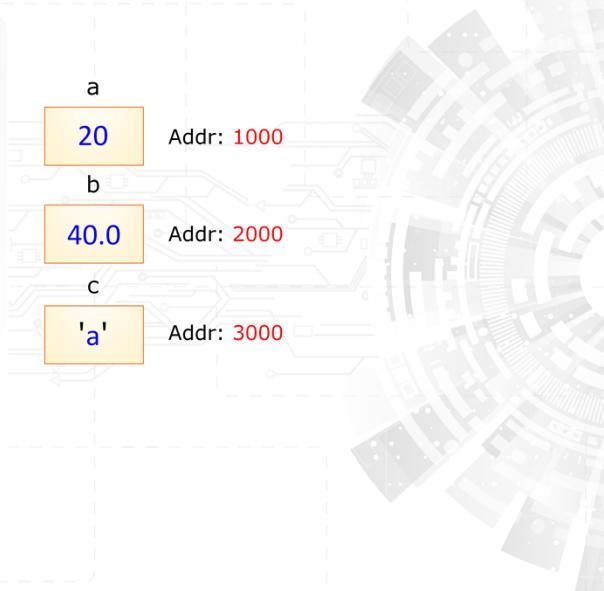
72

And the memory address 3000 is allocated for variable c to store character eh

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

```
int *ptrI;  
float *ptrF;  
char *ptrC;
```



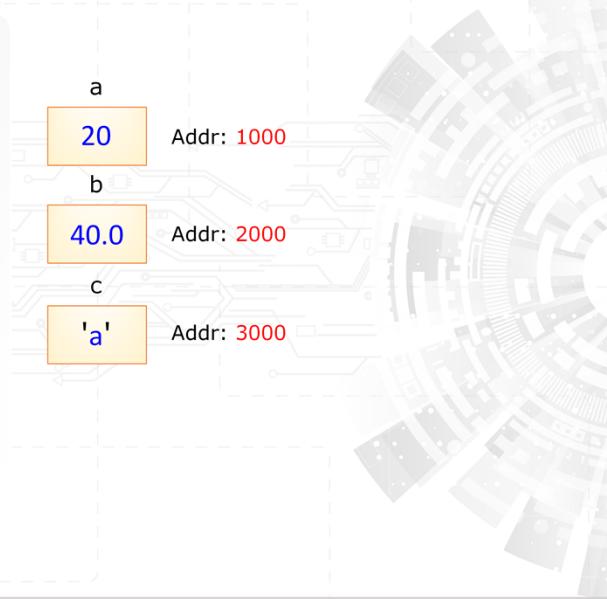
Content Copyright Nanyang Technological University

73

These next 3 statements declared pointer variable. Notice the use of the asterisk sign in the pointer variable declaration?

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;
```



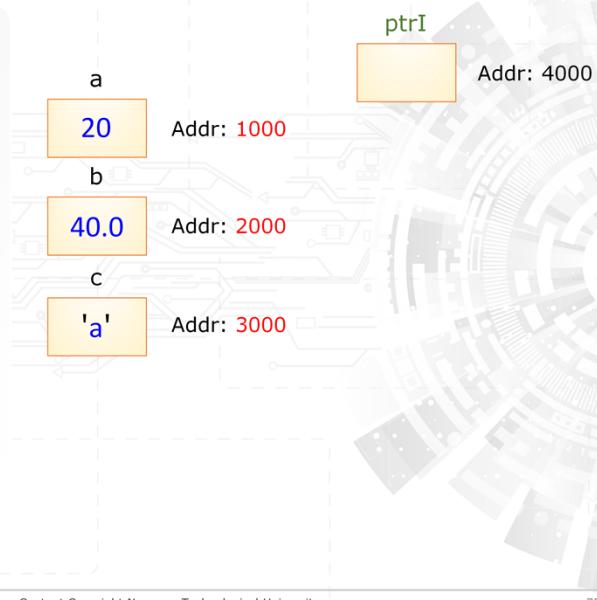
Content Copyright Nanyang Technological University

74

In this statement, the name of the pointer variable is pointer I and is represented by the short form p t r I which we will just call it by pointer i.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;
```



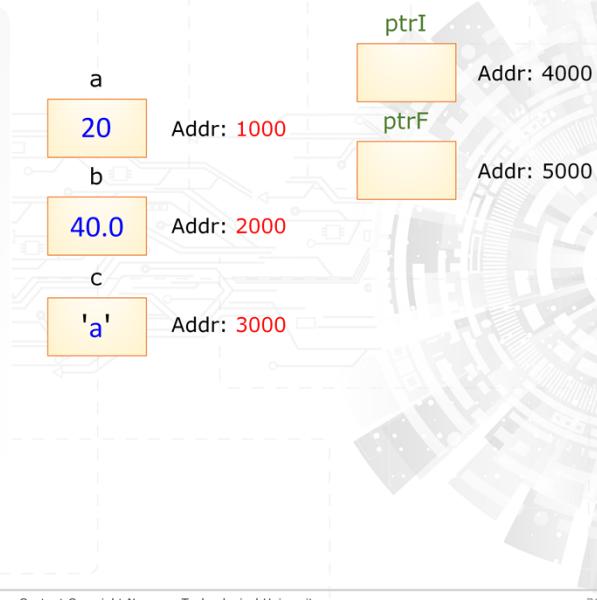
Content Copyright Nanyang Technological University

75

Basically, when this statement is executed, the CPU will just allocate a memory address, say 4000 in this example, for pointer i.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;
```



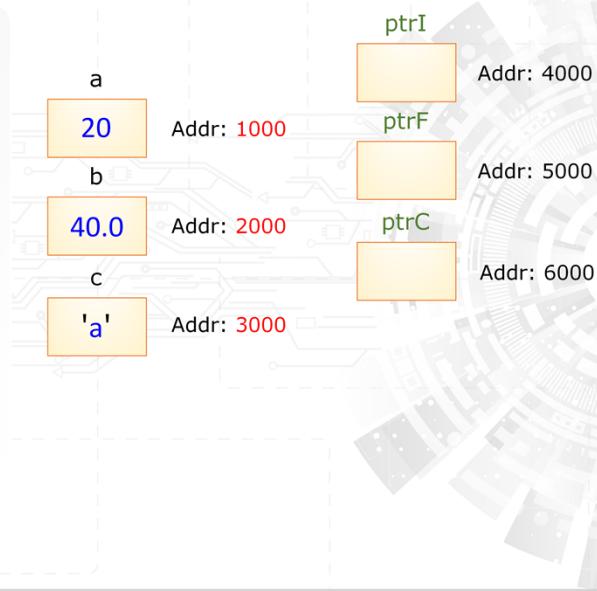
Content Copyright Nanyang Technological University

76

Similarly, pointer F is declared at memory address, say 5000 in this example.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;
```



Content Copyright Nanyang Technological University

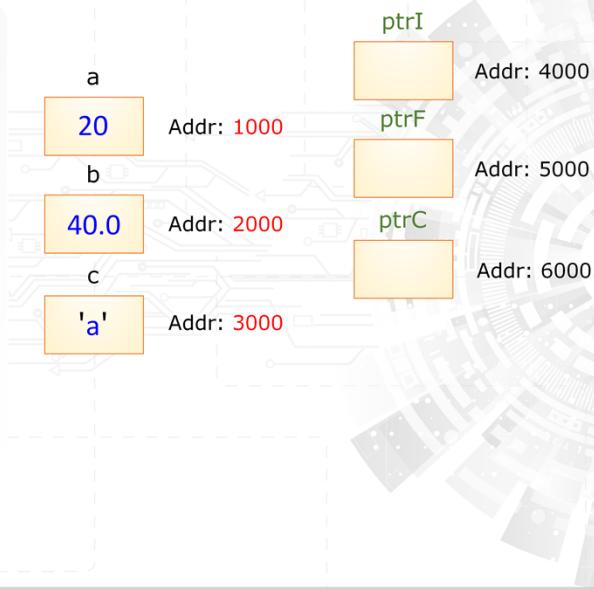
77

Pointer C is declared as well at memory address, say 6000 in this example.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

```
int *ptrI;  
float *ptrF;  
char *ptrC;
```



Content Copyright Nanyang Technological University

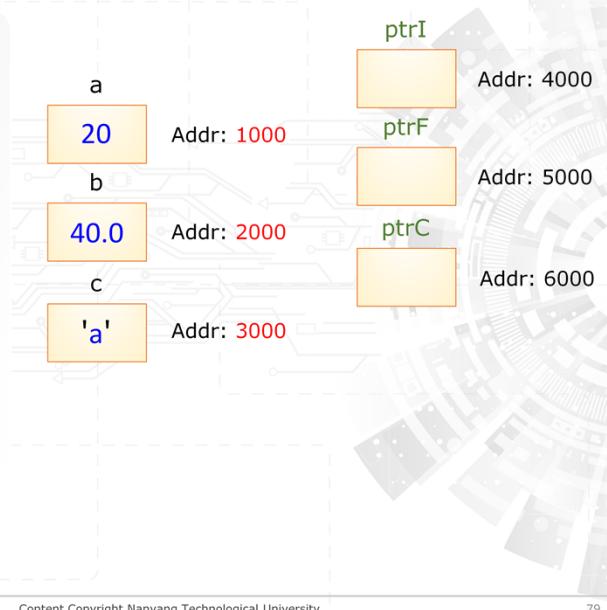
78

Notice that for these 3 statements, pointer variables: pointer I, pointer F and pointer C have been declared but not assigned any value yet.

We will now look into how to assign the values to these pointer variables.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;
```



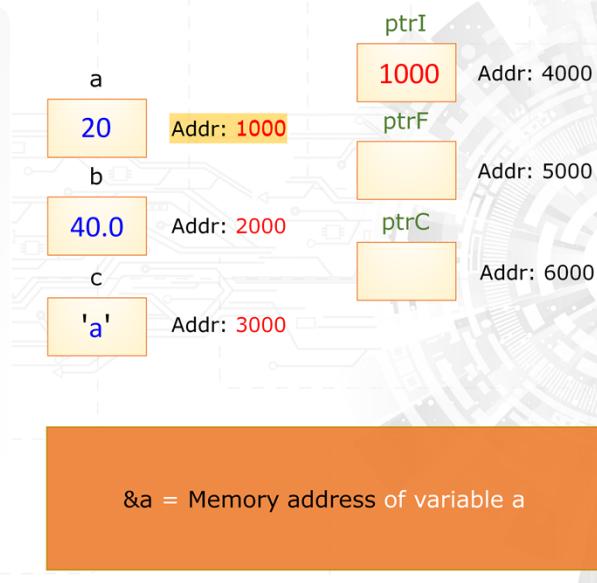
Content Copyright Nanyang Technological University

79

Let's start with pointer I. This statement basically assign the memory address of variable eh to pointer I.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;
```



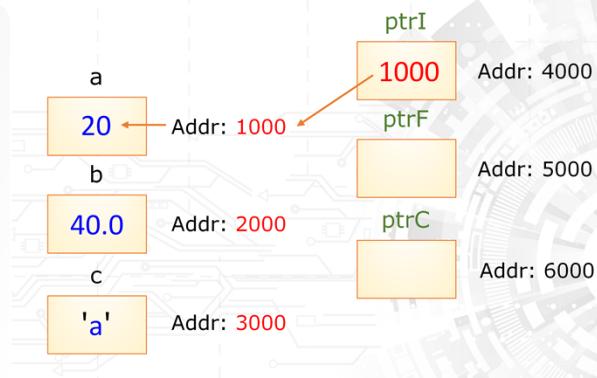
Content Copyright Nanyang Technological University

80

The ampersand sign denotes memory address. So ampersand eh means memory address of variable eh which is 1000 in this example. Thus, the value of 1000 is stored in pointer i.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;
```



`int *ptrI` declares pointer variable `ptrI` to store the memory address which is used to store an integer value.

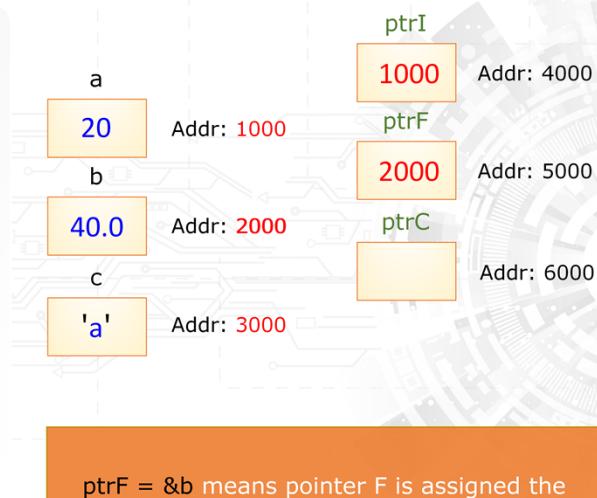
Content Copyright Nanyang Technological University

81

Note that the pointer I was originally declared with statement integer asterisk pointer I which declares pointer variable pointer I to store the memory address that is used to store an integer value.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;
```



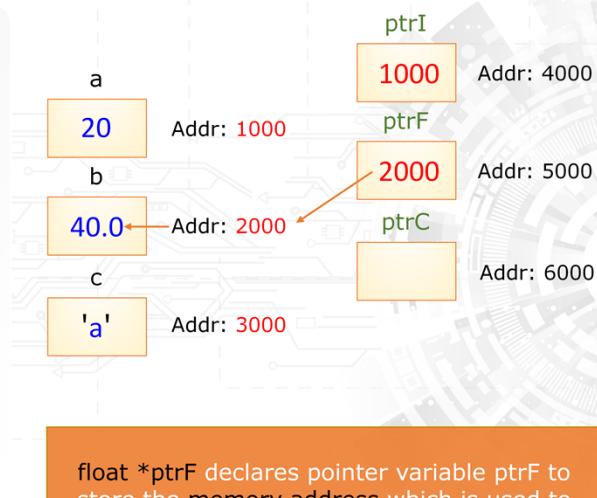
ptrF = &b means pointer *F* is assigned the memory address of variable *b*

Pointer F equals ampersand b

memory address

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;
```



float *ptrF declares pointer variable ptrF to store the memory address which is used to store a float value.

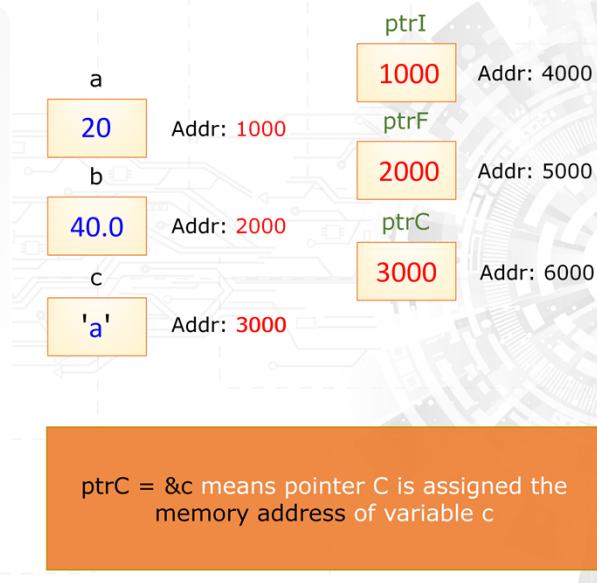
Content Copyright Nanyang Technological University

83

Note that the pointer F was originally declared with statement float asterisk pointer F which declares pointer variable pointer F to store the memory address that is used to store a float value.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;
```



Pointer C equals ampersand c

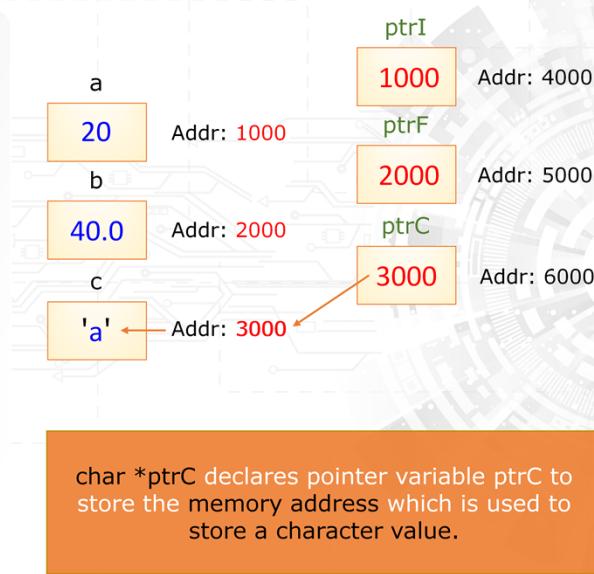
memory address

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

```
int *ptrI;  
float *ptrF;  
char *ptrC;
```

```
ptrI = &a;  
ptrF = &b;  
ptrC = &c;
```



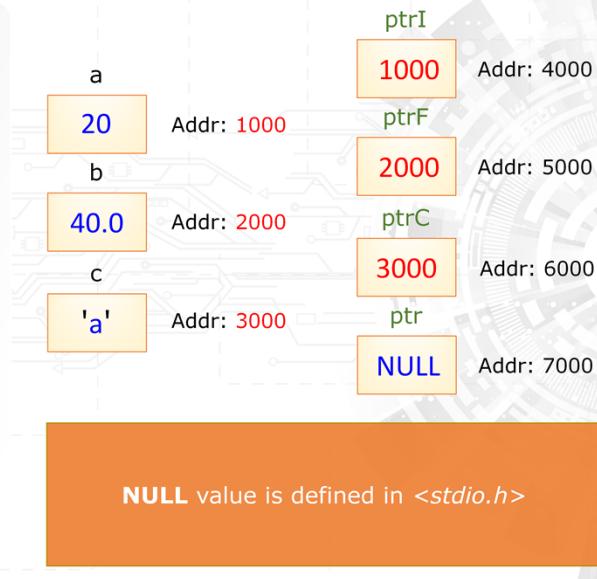
Content Copyright Nanyang Technological University

85

Note that the pointer C was originally declared with statement character asterisk pointer C which declares pointer variable pointer C to store the memory address that is used to store a character value.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



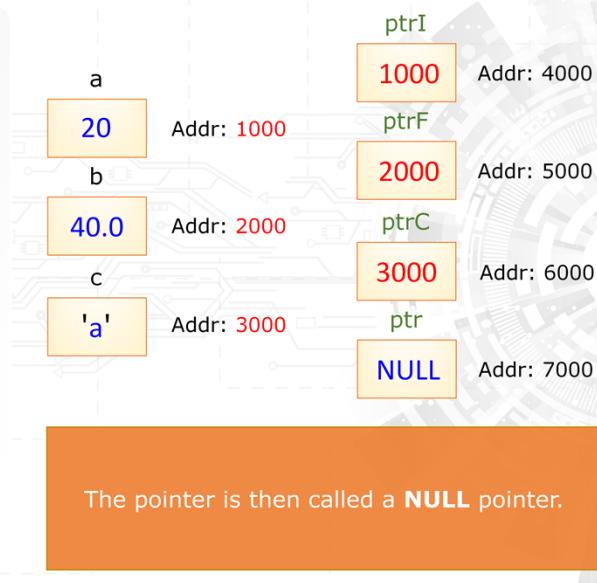
Content Copyright Nanyang Technological University

86

Apart from storing an address value in a pointer variable, we can also store the **NULL** value which is defined in standard eye oh dot aich in a pointer variable.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



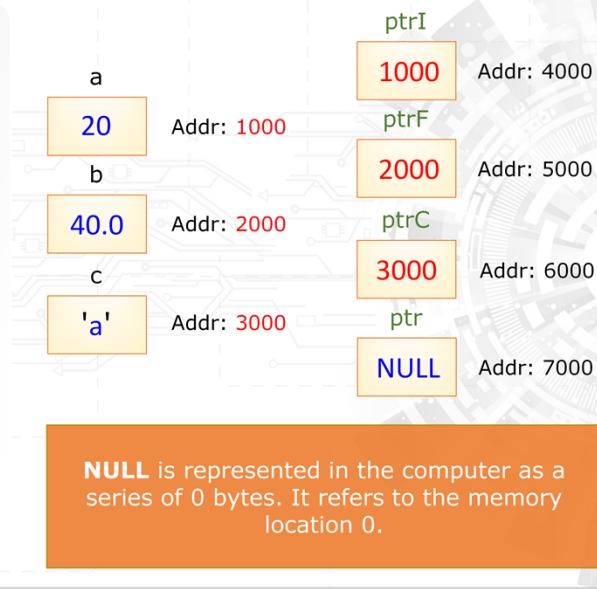
Content Copyright Nanyang Technological University

87

The pointer is then called a **NULL** pointer.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



NULL is represented in the computer as a series of 0 bytes. It refers to the memory location 0.

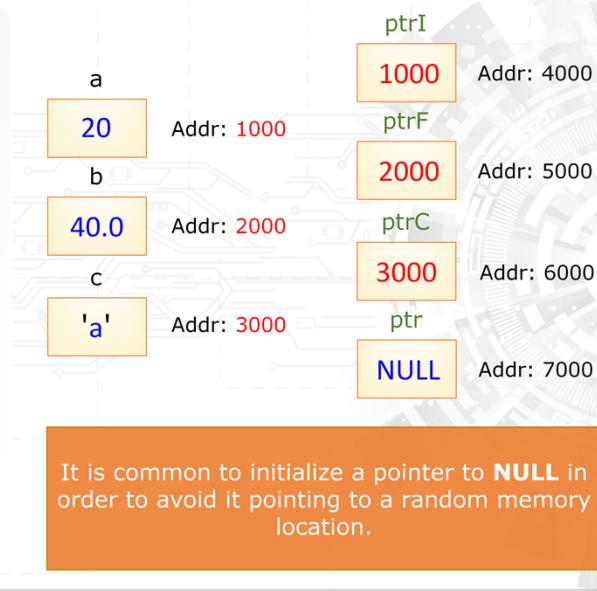
Content Copyright Nanyang Technological University

88

NULL is represented in the computer as a series of 0 bytes. It refers to the memory location 0.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



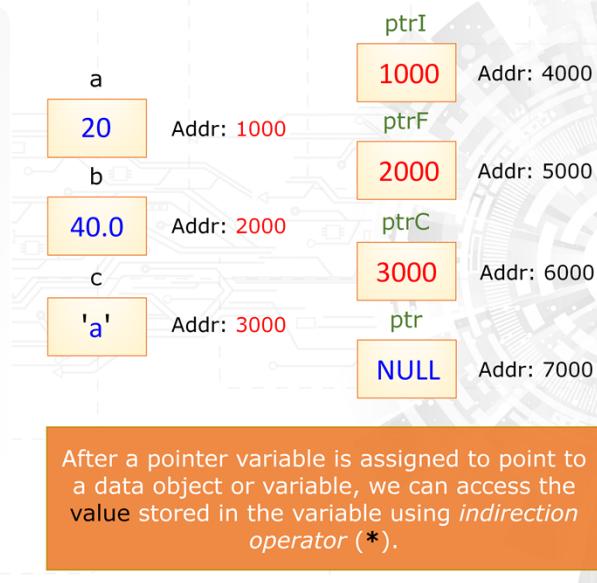
Content Copyright Nanyang Technological University

89

It is common to initialize a pointer to **NULL** in order to avoid it pointing to a random memory location.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



After a pointer variable is assigned to point to a data object or variable, we can access the value stored in the variable using *indirection operator* (*).

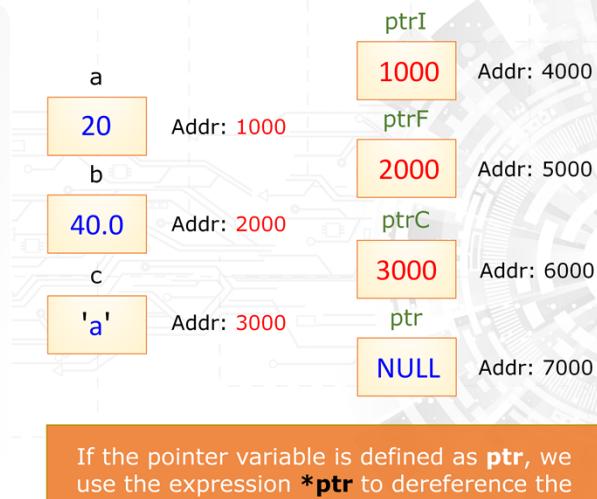
Content Copyright Nanyang Technological University

90

After a pointer variable is assigned to point to a data object or variable, we can access the value stored in the variable using *indirection operator* (*).

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



If the pointer variable is defined as **ptr**, we use the expression ***ptr** to dereference the pointer to obtain the value stored at the address pointed at by the pointer **ptr**.

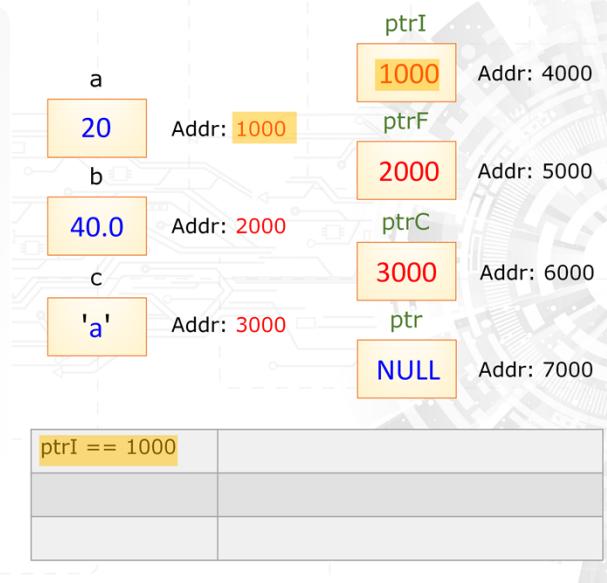
Content Copyright Nanyang Technological University

91

If the pointer variable is defined as **pointer**, we use the expression **asterisk pointer** to dereference the pointer to obtain the value stored at the address pointed at by the pointer. Let's see how this concept is applied in this example.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



Content Copyright Nanyang Technological University

92

After the assignment operations, we will have **pointer I** stores the memory address of the variable **eh**.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```

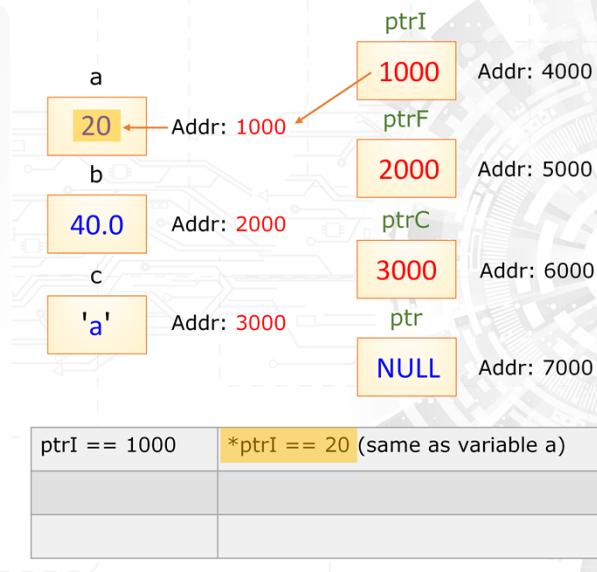
int a=20;
float b=40.0;
char c='a';

int *ptrI;
float *ptrF;
char *ptrC;

ptrI = &a;
ptrF = &b;
ptrC = &c;

int *ptr = NULL;

```



Content Copyright Nanyang Technological University

93

The value of asterisk pointer I is the value stored in memory address of eh which is 20. We can see that asterisk pointer I and variable eh have the same value.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```

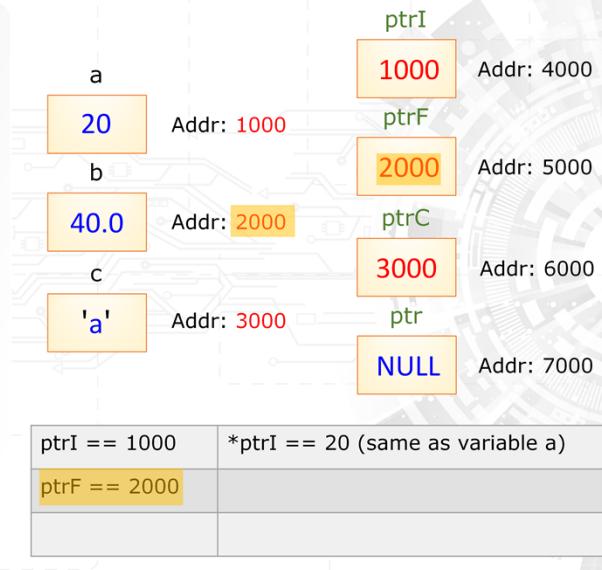
int a=20;
float b=40.0;
char c='a';

int *ptrI;
float *ptrF;
char *ptrC;

ptrI = &a;
ptrF = &b;
ptrC = &c;

int *ptr = NULL;

```



Content Copyright Nanyang Technological University

94

After the assignment operations, we will have **pointer F** stores the memory address of the variable **b**.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```

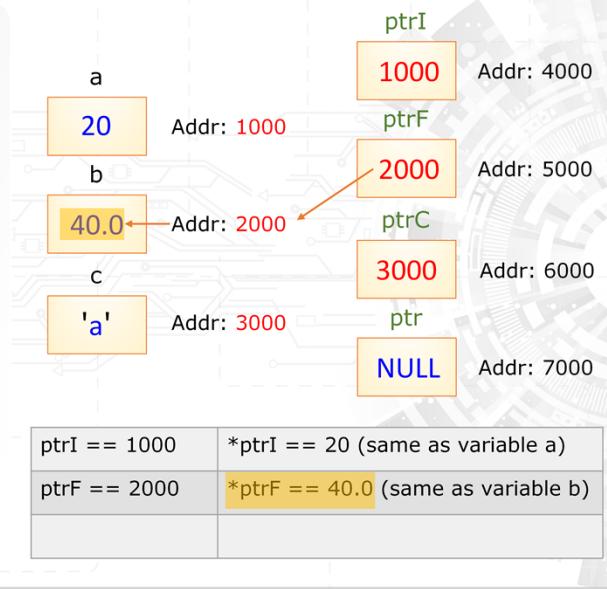
int a=20;
float b=40.0;
char c='a';

int *ptrI;
float *ptrF;
char *ptrC;

ptrI = &a;
ptrF = &b;
ptrC = &c;

int *ptr = NULL;

```



Content Copyright Nanyang Technological University

95

The value of asterisk pointer F is the value stored in memory address of b which is 40.0. We can see that asterisk pointer F and variable b have the same value.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```

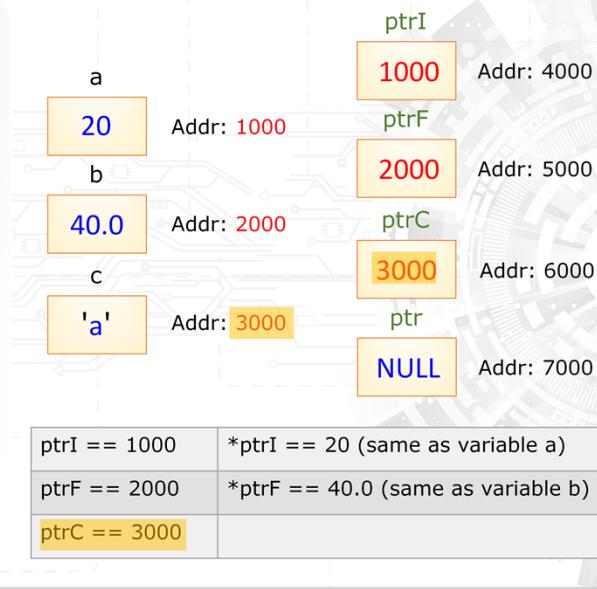
int a=20;
float b=40.0;
char c='a';

int *ptrI;
float *ptrF;
char *ptrC;

ptrI = &a;
ptrF = &b;
ptrC = &c;

int *ptr = NULL;

```



Content Copyright Nanyang Technological University

96

After the assignment operations, we will have **pointer C** stores the memory address of the variable **c**.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```

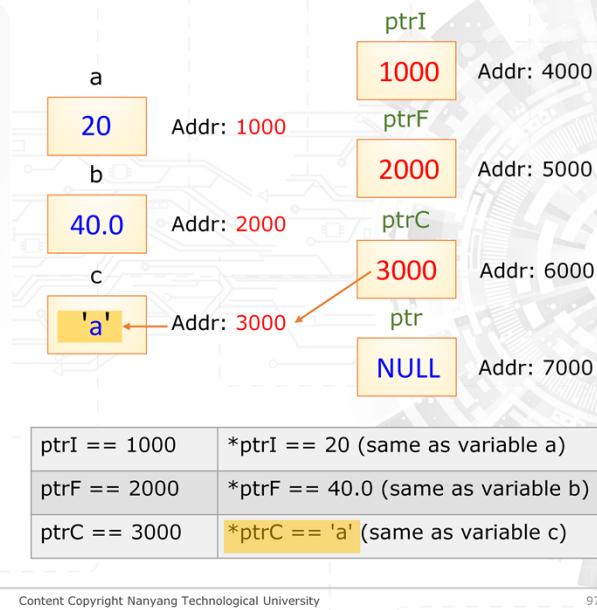
int a=20;
float b=40.0;
char c='a';

int *ptrI;
float *ptrF;
char *ptrC;

ptrI = &a;
ptrF = &b;
ptrC = &c;

int *ptr = NULL;

```



Content Copyright Nanyang Technological University

97

The value of asterisk pointer *C* is the value stored in memory address of *c* which is the character eh. We can see that asterisk pointer *C* and variable *c* have the same value.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

```
int *ptrI;  
float *ptrF;  
char *ptrC;
```

```
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



It means that after the assignment `ptrI = &a;`

<code>ptrI == 1000</code>	<code>*ptrI == 20</code> (same as variable <code>a</code>)
<code>ptrF == 2000</code>	<code>*ptrF == 40.0</code> (same as variable <code>b</code>)
<code>ptrC == 3000</code>	<code>*ptrC == 'a'</code> (same as variable <code>c</code>)

Content Copyright Nanyang Technological University

98

It means that after the assignment **pointer I equals ampersand eh;**

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

```
int *ptrI;  
float *ptrF;  
char *ptrC;
```

```
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



It means that after the assignment `ptrI = &a;` we will be able to retrieve the value of the variable `a` through either

<code>ptrI == 1000</code>	<code>*ptrI == 20</code> (same as variable <code>a</code>)
<code>ptrF == 2000</code>	<code>*ptrF == 40.0</code> (same as variable <code>b</code>)
<code>ptrC == 3000</code>	<code>*ptrC == 'a'</code> (same as variable <code>c</code>)

Content Copyright Nanyang Technological University

99

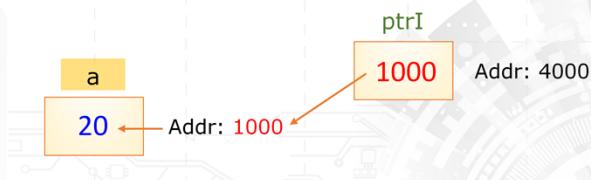
we will be able to retrieve the value of the variable `eh` through either

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

```
int *ptrI;  
float *ptrF;  
char *ptrC;
```

```
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



It means that after the assignment `ptrI = &a;` we will be able to retrieve the value of the variable `a` through either
1. the variable `a` directly; or

<code>ptrI == 1000</code>	<code>*ptrI == 20</code> (same as variable <code>a</code>)
<code>ptrF == 2000</code>	<code>*ptrF == 40.0</code> (same as variable <code>b</code>)
<code>ptrC == 3000</code>	<code>*ptrC == 'a'</code> (same as variable <code>c</code>)

Content Copyright Nanyang Technological University

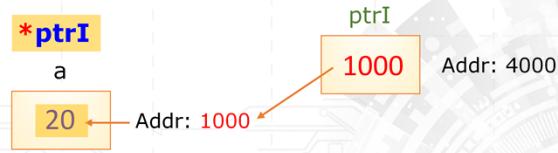
100

the variable `a` directly; or

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

```
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



It means that after the assignment `ptrI = &a;` we will be able to retrieve the value of the variable `a` through either
1. the variable `a` directly; or
2. dereferencing the pointer variable `*ptrI`.

<code>ptrI == 1000</code>	<code>*ptrI == 20</code> (same as variable <code>a</code>)
<code>ptrF == 2000</code>	<code>*ptrF == 40.0</code> (same as variable <code>b</code>)
<code>ptrC == 3000</code>	<code>*ptrC == 'a'</code> (same as variable <code>c</code>)

Content Copyright Nanyang Technological University

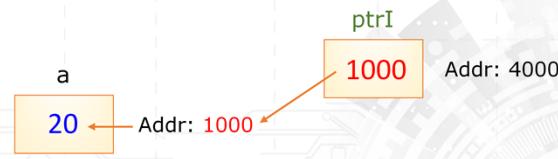
101

dereferencing the pointer variable **asterisk pointer i**.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';
```

```
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



It means that after the assignment `ptrI = &a;` we will be able to retrieve the value of the variable `a` through either
1. the variable `a` directly; or
2. dereferencing the pointer variable `*ptrI`.

<code>ptrI == 1000</code>	<code>*ptrI == 20</code> (same as variable <code>a</code>)
<code>ptrF == 2000</code>	<code>*ptrF == 40.0</code> (same as variable <code>b</code>)
<code>ptrC == 3000</code>	<code>*ptrC == 'a'</code> (same as variable <code>c</code>)

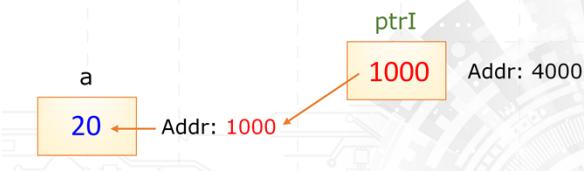
Content Copyright Nanyang Technological University

102

This concept also applies to retrieving of pointer F and pointer C.

ASSIGNING VARIABLE ADDRESS TO POINTER VARIABLES

```
int a=20;  
float b=40.0;  
char c='a';  
  
int *ptrI;  
float *ptrF;  
char *ptrC;  
  
ptrI = &a;  
ptrF = &b;  
ptrC = &c;  
  
int *ptr = NULL;
```



It means that after the assignment `ptrI = &a;` we will be able to retrieve the value of the variable `a` through either
1. the variable `a` directly; or
2. dereferencing the pointer variable `*ptrI`.
Therefore, we can write programs more flexibly by using pointer variable.

<code>ptrI == 1000</code>	<code>*ptrI == 20</code> (same as variable <code>a</code>)
<code>ptrF == 2000</code>	<code>*ptrF == 40.0</code> (same as variable <code>b</code>)
<code>ptrC == 3000</code>	<code>*ptrC == 'a'</code> (same as variable <code>c</code>)

Content Copyright Nanyang Technological University

103

Therefore, we can write programs more flexibly by using pointer variable.

SUMMARY

We have completed the lesson on Pointer Variables (Part 1 of 4) and you have learnt to:

- Explain the use of pointer variables
- Apply pointer variables
- Apply indirection operator
- Apply pointer variables by assigning variable address to pointer variables

We have completed the lesson on Pointer Variables (Part 1) and you have learnt the points listed. In the next lesson, we will be learning how these points work in proper programming codes with an example.