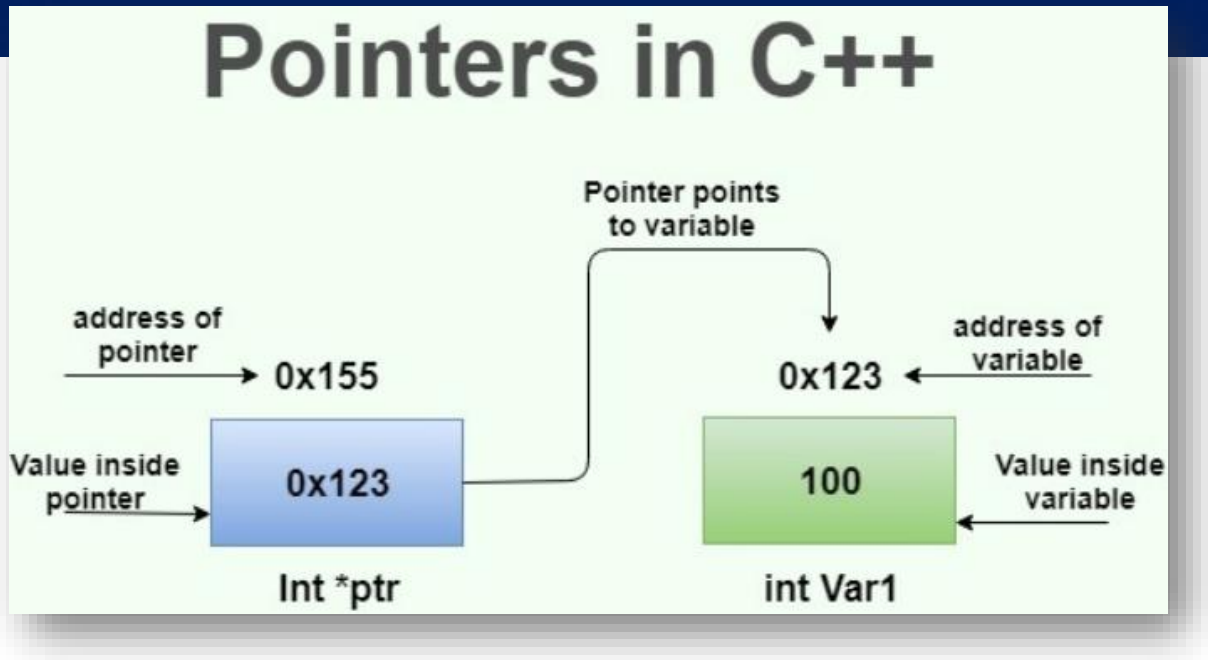
DATA STRUCTURE AND PROGRAMMING II



Pointer

Write program in C++ using pointer

Remark: Pointer is a variable that store address of another variable of the same data type.

Lecture overview

❑ Overall lectures

1. Introduction to algorithm
2. Basic data types and statements
3. Control structures and Loop
4. Array
5. Data structure
6. Sub-programs
7. Recursive
8. File IO
9. *Pointers*
10. Linked Lists
11. Stacks and Queues
12. Sorting algorithms
13. Trees

C++

Outline

- What is pointer?
- What are the advantages of using pointer?
- How to use pointer
- Examples

Introduction

❑ Computer Memory

- To understand pointers, you should have knowledge about address in computer memory

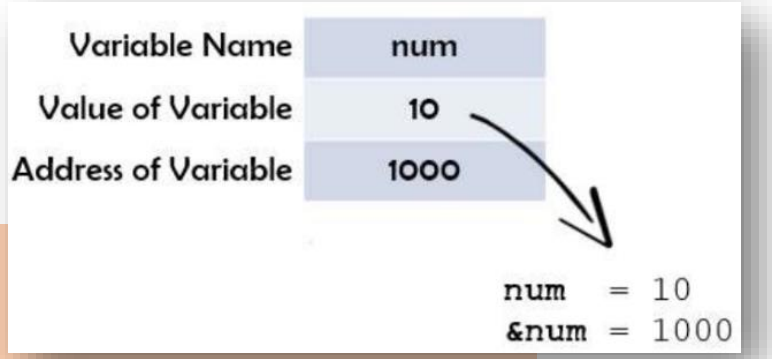
- A computer memory location has an address and holds a content (value)
- The address is a numerical number (expressed in hexadecimal)
- An integer value consumes 4 bytes of memory

Memory	
Address	Value
0045FFCBI89	abc
0276GGHBC00	abcd

Introduction

❑ Computer Memory

- Each variable we create in the program has a location in the computer's memory
- The value of the variable is stored in the assigned location
- To know where the data of normal variable is stored, we use operator **&**
  - **&** gives the address occupied by a variable

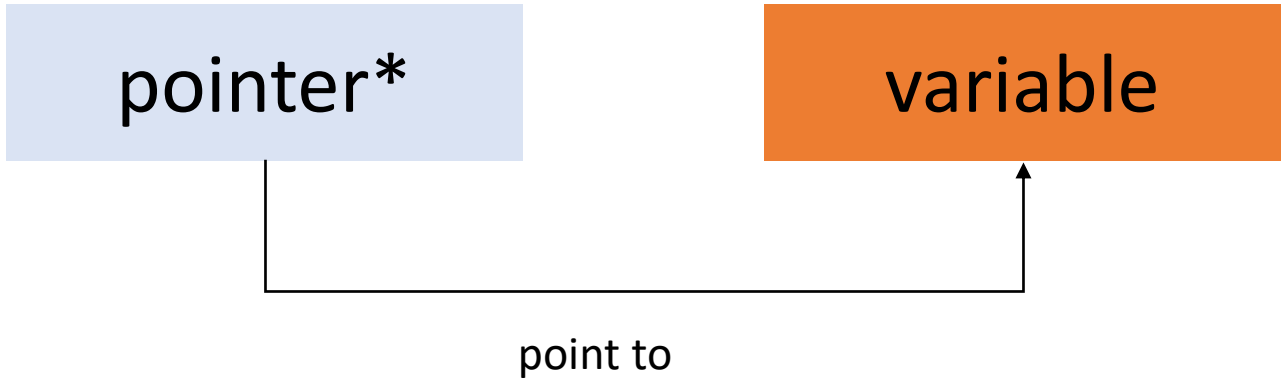


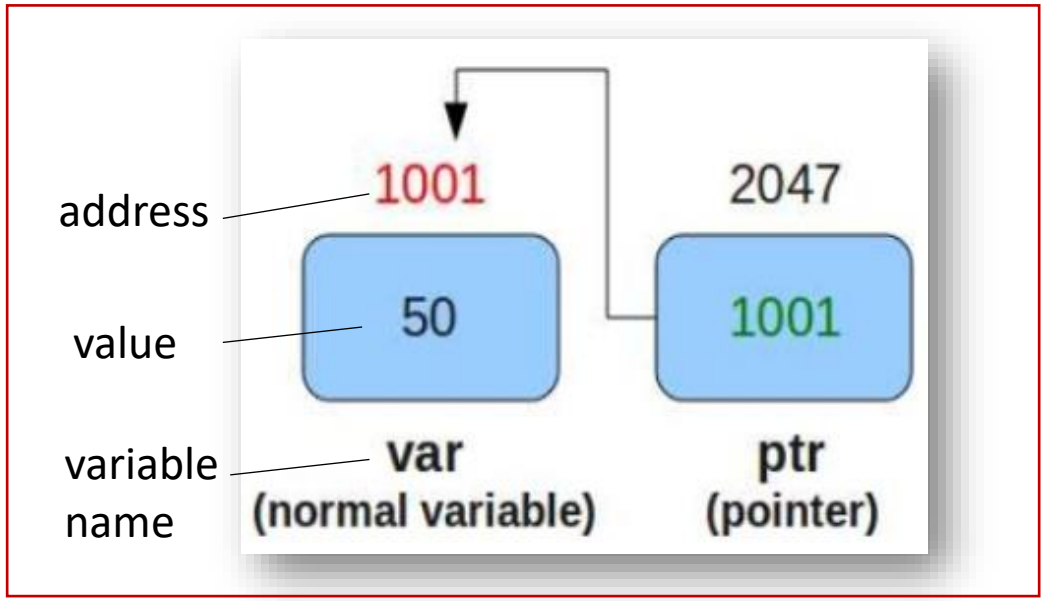
- Example:
  - If ***num*** is a variable, then ***&num*** gives the address of that variable

Introduction

❑ What is pointer?

- A pointer is a variable that holds the memory address of another variable of the same type.
- Pointers are used to access the memory address and values at that address.



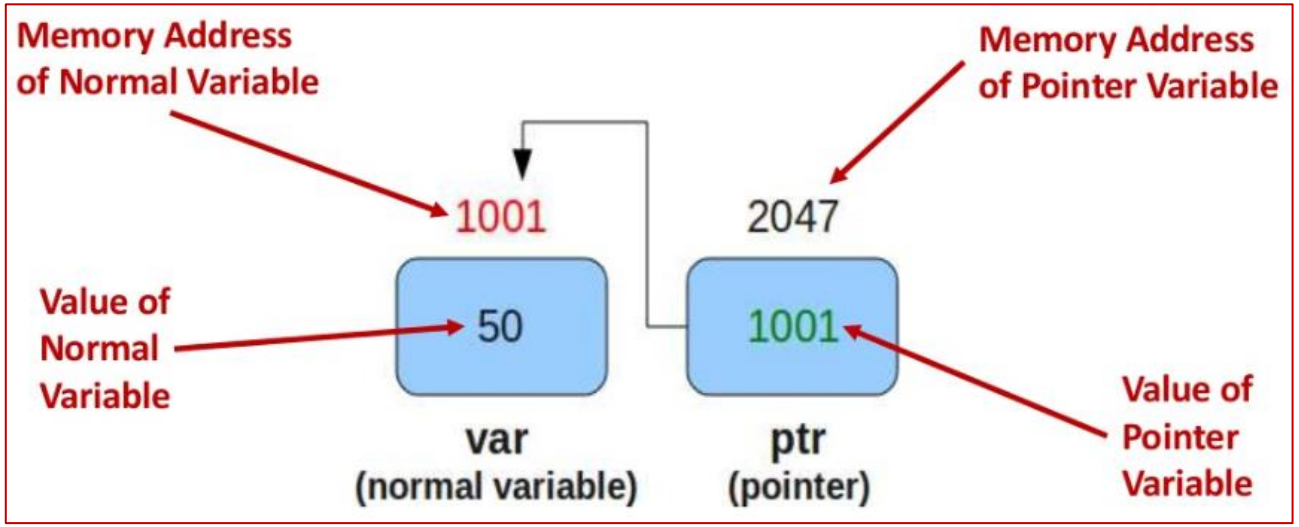


An example of a pointer variable pointing to a normal variable

Pointer Vs. Normal variable

❑ Remark

- A normal variable is used to store value, while *a pointer variable is used to store address (reference) of another variable*
- Pointers are representation of addresses
- We can have a pointer to any variable type



Advantages of using pointer?

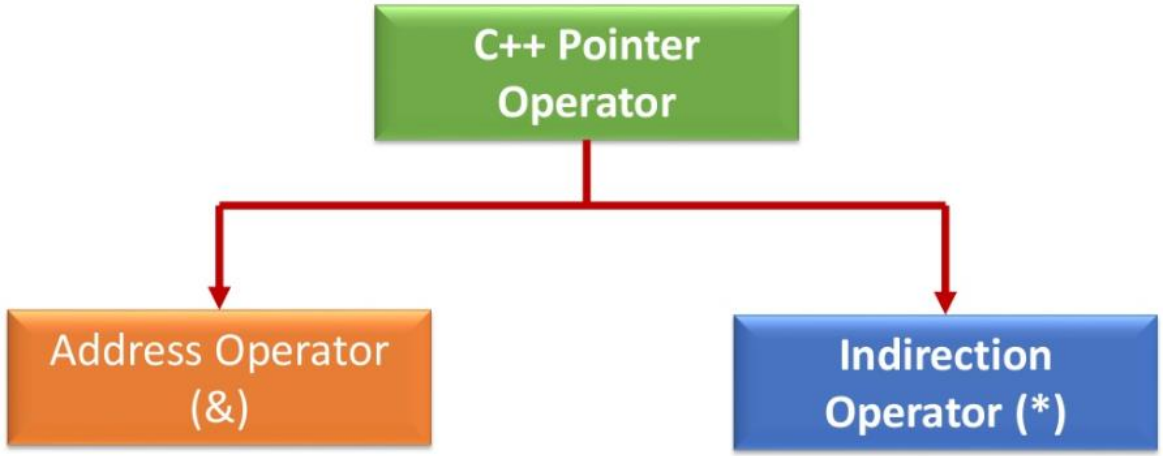
❑ Some main advantages

1. Use less memory
   - Dynamic memory allocation
2. Program runs faster
   - Increase execution speed and reduce execution time
3. Efficient when work with array, structure, list, stack, queue, …
4. Provide another way to access array element
5. Instead of copying data, pointer just point to an existing data
6. A function can return more than one value by passing via function argument

Pointer Operator

❑ What?

- Two operators when work with pointer
  - Address operator (**reference** operator)
    - It uses **&**
    - It returns memory address
  - Indirection operator (**deference** operator or value operator)
    - It uses ***
    - It returns value



Pointer Declaration

❑ Syntax

A pointer is a variable that must be defined by specifying the type of variable pointed, as follows

```
var *nameOfPointer : type
```

The type of variable pointed can be a primary type (such as integer, character ...) or a complex type (such as structured type ...)

```
structure students
          name: string
          age: integer
end structure
```

```
var *a: integer
    *c: character
    *s: string
    *stu: students
```

Pointer Initialization

❑ Syntax

var *a: integer a ← NULL	var *c, d: character c ← &d

Access to Pointer Variable

❑ Syntax

- After (and only after) having declared and initialized a pointer, it is possible to access the contents of the memory address pointed by the pointer by the operator '*'
- Syntax: *nameOfPointer

```
var *a, b: integer
a ← &b
b ← 10
*a ← *a + b
write(*a, b)
```

- a: pointer (stores an address)
- * a: pointer variable (as integer variable or in this case it is equivalent to variable **b**)

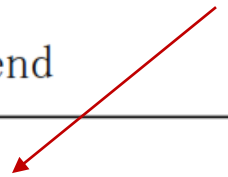
Example 1

❑ Not using pointer

- A function that exchanges the values of two variables (How does it work?)

```
function exchange(v1: integer, v2: integer)
begin
var tmp: integer
    tmp ← v1
    v1 ← v2
    v2 ← tmp
end
```

```
main program
begin
    var a,b: integer
        a ← 1
        b ← 2
        exchange(a,b)
        write(a,b)
end
```



What are the values of **a** and **b** here?

a is 1, b is 2

Example 2:

❑ Using pointer

- A function that exchanges the values of two variables (How does it work?)

```
function exchange(*v1: integer, *v2: integer)
begin
var tmp: integer
    tmp ← *v1
    *v1 ← *v2
    *v2 ← tmp
end
```

```
main program
begin
    var a,b: integer
        a ← 1
        b ← 2
        exchange(&a, &b)
        write(a, b)
end
```

What are the values of **a** and **b** here?

a is 2, b is 1

Pointer and Array

- The name of an array represents the address of its first element **&array [0]**

```
var A[10], *P: integer
        P ← A
```

- If P points to any element of an array, then P + 1 points to the next element
- More generally:

  (P+i) points to the i^{th} element behind P and

  (P-i) point to the i^{th} element before P

Pointer and Array

❑ Remark

- Suppose we have variables
  - `Var arr[10]: integer`
  - `Var *p:integer`

- Array name `arr` represents the address of the first elements of this array `(&arr[0])`
- We can say
  - `p = arr;` // `p` point to the first element `(arr[0])` in the array
- When a pointer points to an array, the value of the pointer is the first array element
  - `write(*p)`

NOTE

❑ Reference (&) Vs. Deference (*) operator

- &: to get address of any variable
- *: to get value at the address that the point stores

Example:

- If an integer variable, say n, is stored in memory address 0xf1bd23, and n contains a value of 5.

Then:

Reference operator **&n** gives the value of 0xf1bd23
Deference operator ***n** gives the value of 5

Q&A

Example 1: Using C++

- Suppose we have program as follows

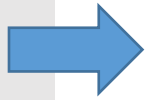
```
#include<iostream>
using namespace std;
int main(){
        int num=10;
        int *ptr;
        ptr = &num;

        cout<<“num=”<<num<<endl;
        cout<<“Address of variable num is: ”<<&num<<endl;
        cout<<“Value in variable pointer ptr is: ”<<ptr<<endl;
        cout<<“*ptr=”<<*ptr<<endl;      //Value in that address
}
```

Example 2: Using C++

```
#include<iostream>
using namespace std;
int main(){
        int *pc, c;
        c=5;

        cout<<“Address of c: ”<<&c<<endl;
        cout<<“Value of c: ”<<c<<endl;
        pc = &c;
        cout<<“Address that pc holds: ”<<pc<<endl;
        cout<<“Value of address that pc holds: ”<<*pc<<endl;
        c = 11;
        cout<<“Address that pc holds: ”<<pc<<endl;
        cout<<“Value of address that pc holds: ”<<*pc<<endl;
        *pc = 2;
        cout<<“Address of c: ”<<&c<<endl;
        cout<<“Value of c: ”<<c<<endl;
}
```



Output

```
Address of c: 0x6dfef8
Value of c: 5
Address that pc holds: 0x6dfef8
Value of address that pc holds: 5
Address that pc holds: 0x6dfef8
Value of address that pc holds: 11
Address of c: 0x6dfef8
Value of c: 2
```

Pointer Examples

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

main(){
    int num;
    int n,p,q;

    num = 20;
    n = 1;
    p = 2;
    q = 3;

    cout<<&num<<endl;
    cout<<&n<<endl;
    cout<<&p<<endl;
    cout<<&q<<endl;
}
```

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

main(){
    int a[5];
    a[0] = 2;
    a[1] = 8;
    int *p1;

    p1 = &a[0];

    cout<<&a[0]<<endl;
    cout<<&a[1]<<endl;

    cout<<p1<<endl;
}
```

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

main(){
    int *a;
    int b=10;

    a = &b;

    cout<<b<<endl;
    *a = 15**a;
    cout<<b<<endl;
}
```

Write program in C++ using pointer

Remark: Pointer is a variable that store address of another variable of the same data type.

Q&A