

Programming Technique II – SCJ1023

Pointers

Associate Prof. Dr. Norazah Yusof

What is a pointer?

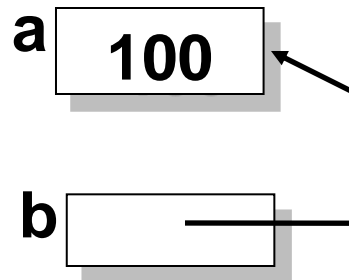
- A pointer is a derived data type
 - A data type built from one of the standard data type.
- Pointer variables contain memory address of a variable
 - Indirectly references a value.

What is a pointer variable?

- A *pointer variable* is a special type of variable that holds the *address of another variable*.

- Example:


Variable `b` in is a pointer variable which is containing the address of variable `a`.



What is address operator in C++?

- & is the address operator in C++.
 - Is used to *get the address* of a variable.
 - Provides a pointer constant to any named location in memory.
- Example 1: Display the address of an integer variable.

```
int Nom;  
cout << &Nom;
```

Nom
&Nom 

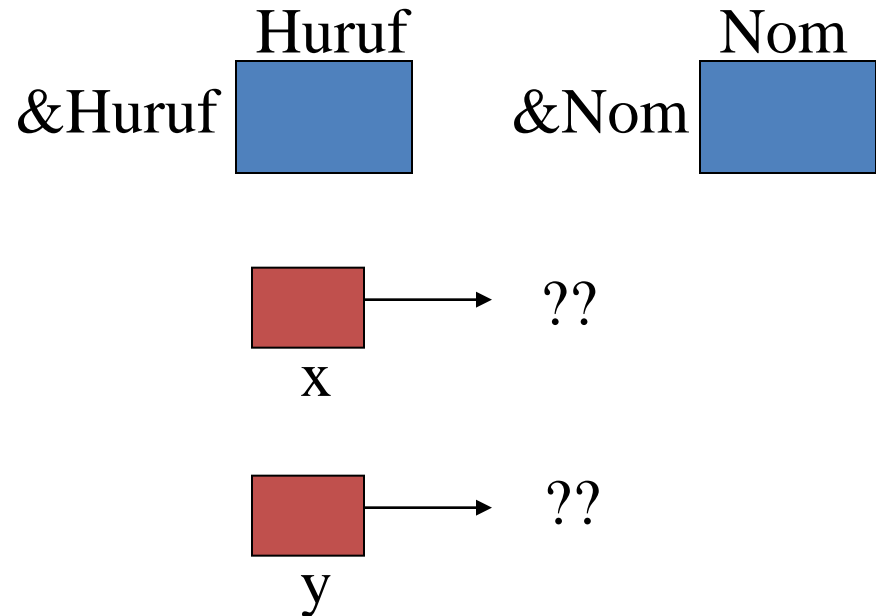
What is dereferencing in C++?

- The *dereferencing operator*, `*`, is used to *get the variable* that a pointer variable is pointing to.
- This operator can only be applied to pointer variables (not to ordinary variables).

Example of declaration of pointer variables

- Pointer variable stores the address of a variable.

```
char Huruf;  
int Nom;  
char * x;  
int * y;
```



Example of declaration of pointer variables

C++ declarations:

```
char Huruf;
```

```
int Nom;
```

```
char * x;
```

```
int * y;
```

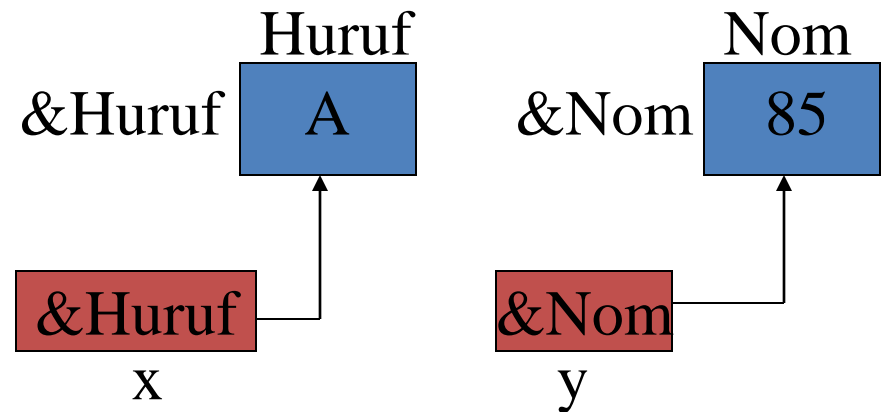
Contents of variables:

```
Huruf = 'A';
```

```
Nom = 85;
```

```
x = &Huruf;
```

```
y = &Nom;
```

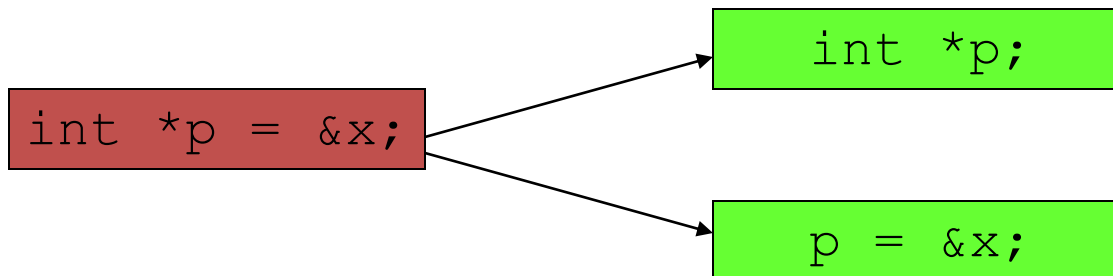


Example of initialization of pointer variables

C++ declarations:

```
int x=25;
```

```
int *p = &x;
```



Example of assigning of pointer to another pointer

C++ declarations:

```
int x = 25, y=100;  
int *ptr1;  
int *ptr2 = &y;
```

Assigning values from pointer variables:

```
ptr1 = &x;
```

Assigning pointer address to another pointer

```
ptr2 = ptr1;
```

Output

```
The value in x is 25  
The address of x is 0x0012ff88  
The address in ptr1 is 0x0012ff88  
The address in ptr2 is 0x0012ff88
```

Accessing variables through pointer

- To access to the pointed variable, use * as the indirection operator.
- Various operations can be done using indirection operator.
 - Assign value
 - Input/output operations
 - Arithmetic operations
 - Relational/logical expressions

Relationship between Arrays and Pointers

- Array name is the starting address of an array

```
int vals[] = {4, 7, 11};
```

Example of starting address of `vals`:

```
0x4a00
```

4	7	11
---	---	----

```
cout << vals;           // displays  
                        // 0x4a00  
cout << vals[0];       // displays 4
```

Relationship between Arrays and Pointers

- Array name can be used as a pointer constant:

```
int vals[] = {4, 7, 11};  
cout << *vals; // displays 4
```

- Pointer can be used as an array name:

```
int *valptr = vals;  
cout << valptr[1]; // displays 7
```

Array Access

- Array elements can be accessed in many ways:

Array access method	Example
array name and []	<code>vals[2] = 17;</code>
pointer to array and []	<code>valptr[2] = 17;</code>
array name and subscript arithmetic	<code>*(vals + 2) = 17;</code>
pointer to array and subscript arithmetic	<code>*(valptr + 2) = 17;</code>

Pointer Arithmetic

- Operations on pointer variables:

Operation	Example
++, --	<pre>valptr++; // points at 7 valptr--; // now points at 4</pre>
+, - (pointer and int)	<pre>cout << *(valptr + 2); // 11</pre>
+=, -= (pointer and int)	<pre>valptr = vals; // points at 4 valptr += 2; // points at 11</pre>
- (pointer from pointer)	<pre>cout << valptr - val; // difference // (number of ints) between valptr // and val</pre>

Comparing Pointers

- Relational operators (<, >=, etc.) can be used to compare addresses in pointers
- Comparing addresses in pointers is not the same as comparing contents pointed at by pointers:

```
if (ptr1 == ptr2) // compares
                  // addresses
if (*ptr1 == *ptr2) // compares
                   // contents
```


Pointers as Function Parameters

- A pointer can be a parameter
- Works like reference variable to allow change to argument from within function
- Requires:

1) asterisk * on parameter in prototype and heading

```
void getNum(int *ptr); // ptr is pointer to an int
```

2) asterisk * in body to dereference the pointer

```
cin >> *ptr;
```

3) address as argument to the function, use &

```
getNum(&num); // pass address of num to getNum
```