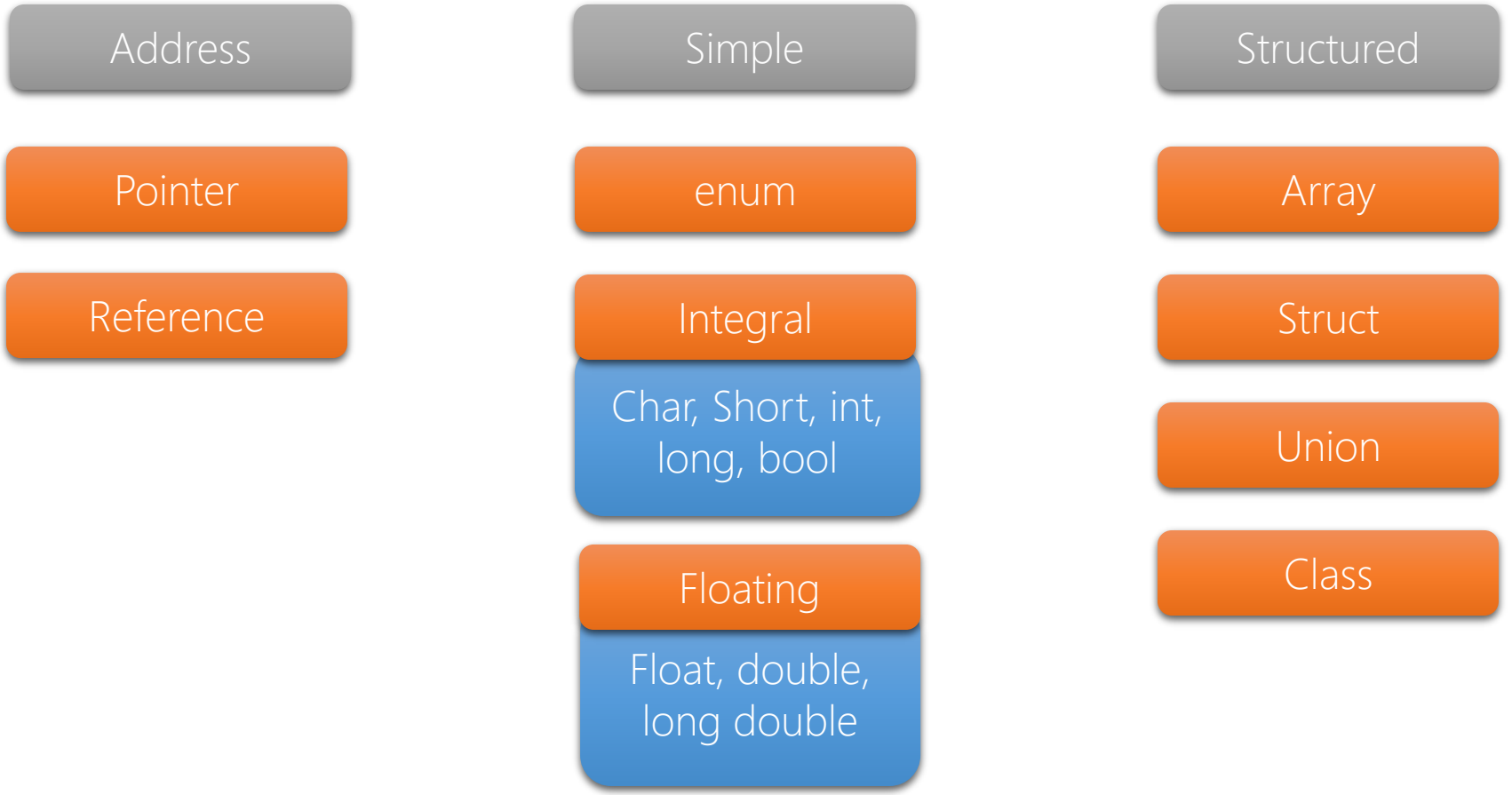
C++

PROGRAMMING LANGUAGE

L06-POINTERS

Mohammad Shaker
mohammadshaker.com
@ZGTRShaker
2010, 11, 12, 13, 14

C++ data types


Address
Pointer
Reference
Simple
enum
Integral
Char, Short, int, long, bool
Floating
Float, double, long double
Structured
Array
Struct
Union
Class

Pointers

Pointers

powerful, difficult

Pointers

- Powerful feature in C++, but one of the most difficult to master
- Using Dynamic objects
  - Can survive after the function ends in which they were allocated
  - Allow flexible-sized arrays and lists
  - Lists, Trees, stacks, Queues
  - Managing big sized, large objects throw passing them into functions

- Note:
  - Can declare pointer to any data type
- In general the type of the pointer must match the type of the data it's set to point to

Pointers

- Pointer variable contain an “address” rather than a data “value”

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

void main(void)
{
        int *ptr; // initialize a pointer to an integer object
}
```

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

void main(void)
{
        float *ptr; // initialize a pointer to an float object
}
```

Pointers

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

void main(void)
{
        int *ptr; // initialize a pointer to an integer object
}
```

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

void main(void)
{
        int* ptr; // initialize a pointer to an integer object
}
```

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

void main(void)
{
        int * ptr; // initialize a pointer to an integer object
}
```

Pointers

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

void main(void)
{
        int *ptr1, *ptr2; // ptr1, ptr2 are pointers
}
```

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

void main(void)
{
        int *ptr1, p;
        // ptr1: pointer to int
        // p: int
}
```

Pointers

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

void main(void)
{
        int *ptr1, p;
        // ptr1: pointer to int
        // p: int
}
```

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

void main(void)
{
        int* ptr1, p;
        // ptr1: pointer to int
        // p: int
}
```

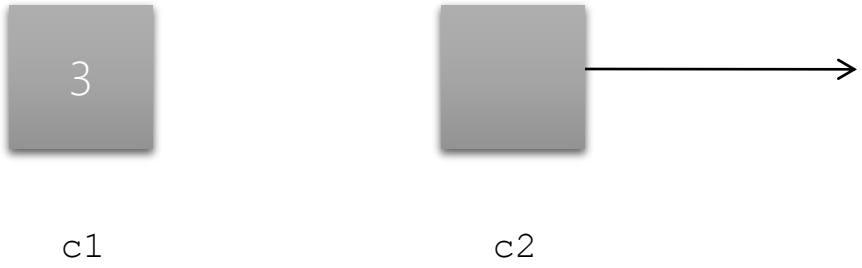
Pointers

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

typedef int* IntPtr;


void main(void)
{
          IntPtr Ptr1,Ptr2; // now, Ptr1,Ptr2 both are pointers to integer
}
```

```
          int *ptr, c;         // ptr: pointer, c: integer
          int*  ptr,c;                  // ptr: pointer, c: integer
          int(*  ptr),c;       // ptr: pointer, c: integer
          int       (*ptr),c; // ptr: pointer, c: integer
          int c, *p;                    // ptr: pointer, c: integer
          (int*) ptr,c;        // compiler error


          int*    ptr;                  // ptr: pointer
          int  *  ptr;                  // ptr: pointer
          int    *ptr;                  // ptr: pointer
```

Pointers

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


void main(void)
{
        int c1=3, *c2; // c1: integer, c2: pointer to int
}
```



c2 is not pointing to anything, because it hasn't been initialized, thus called, "null" pointer

Pointers

- " The address of " operator
  - &

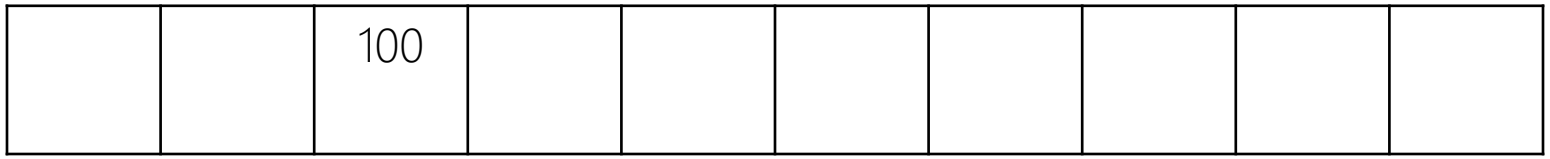
```
#include <iostream>
using namespace::std;


void main(void)
{
        int c1;
        c1 = 3;
        cout << "The value of variable c1 = " << c1 << endl;
        cout << "The address of variable c1 in memory = " << &c1 << endl;
}
```

```
The value of variable c1 = 3
The address of variable c1 in memory = 0024EC64
Press any key to continue
```

Pointers

```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100 ;
        int *ptr = new int;
        *ptr = i ;
}
```

memory 1024

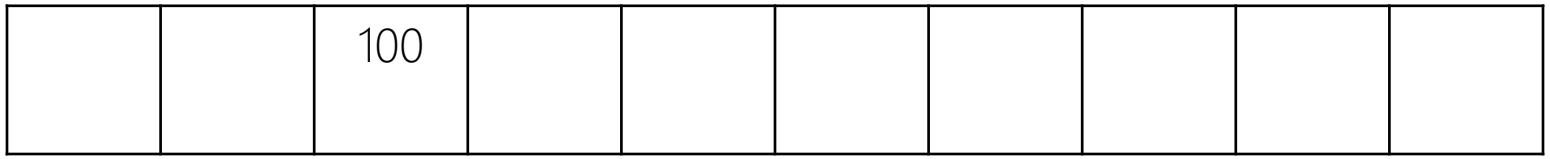


- A pointer is also a "variable"
  - So it also has its own memory address

Pointers

```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100 ;
        int *ptr = new int;
        *ptr = i ;
}
```

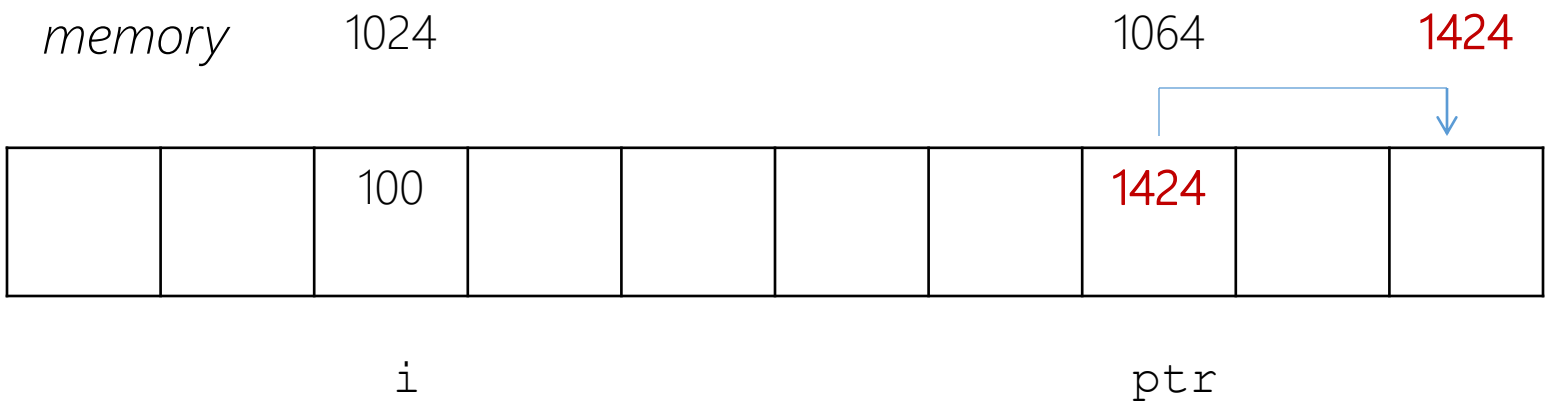
memory 1024



- A pointer is also a "variable"
  - So it also has its own memory address

Pointers

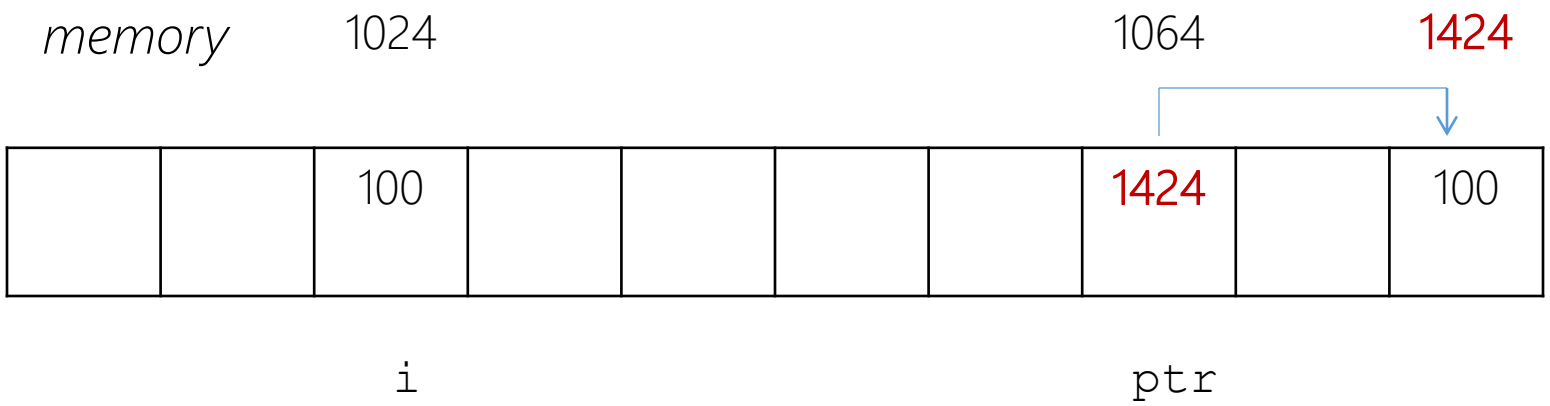
```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100 ;
        int *ptr = new int;
        *ptr = i ;
}
```



- A pointer is also a "variable"
  - So it also has its own memory address

Pointers

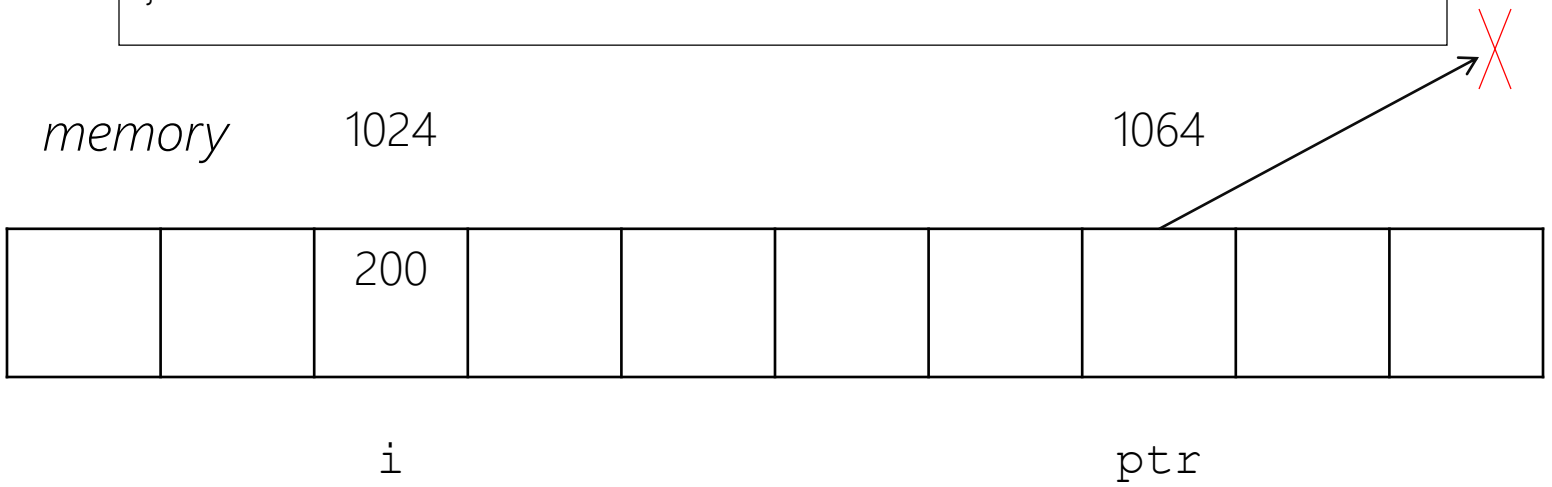
```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100 ;
        int *ptr = new int;
        *ptr = i ;
}
```



- A pointer is also a "variable"
  - So it also has its own memory address

Pointers

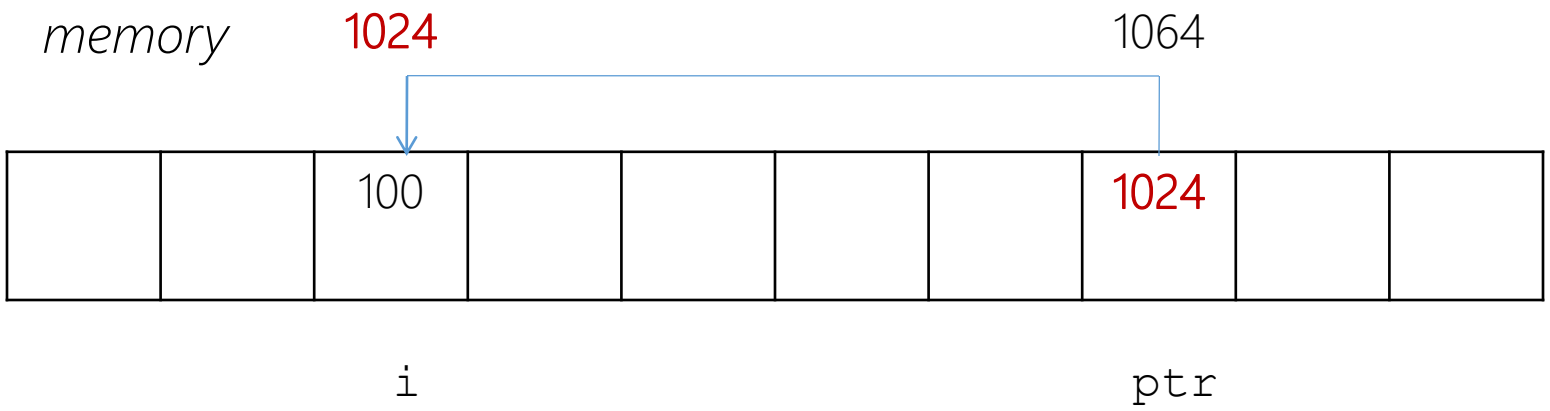
```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 200 ;
        int *ptr ;
        *ptr = 200 ;
}
```



Runtime Error

Pointers

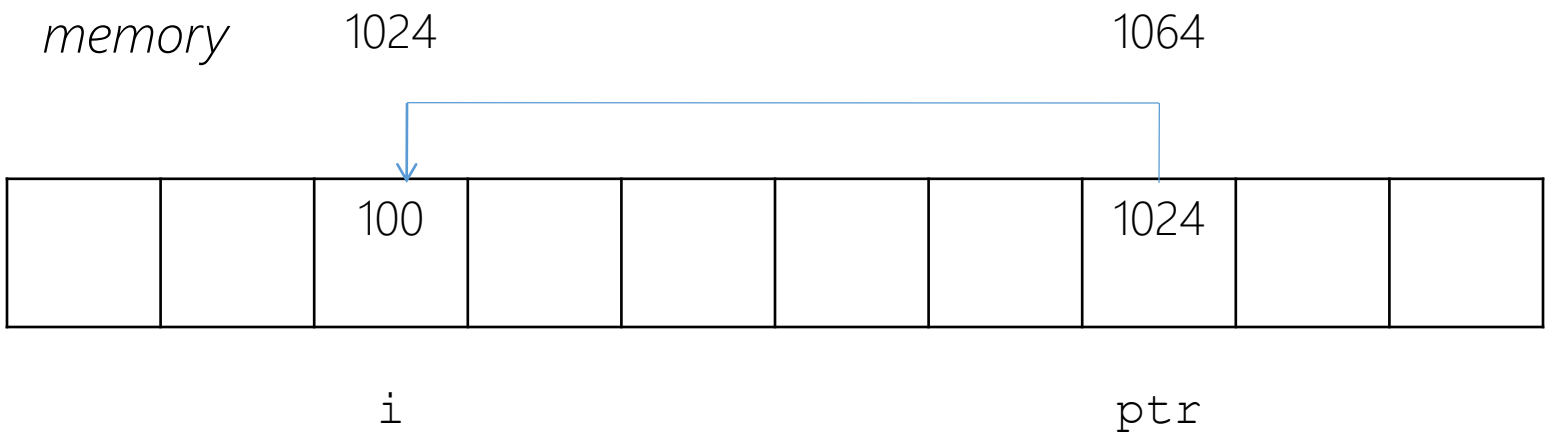
```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100 ;
        int *ptr ;
        ptr = &i ;
}
```



The "*pointer*" now has the "*address*" of the "*variable*"

Pointers

```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100 ;
        int *ptr = &i ;
        // initialize at definition time
}
```



Pointers

```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr;
        ptr = &i;
}
```

```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = &i;
}
```

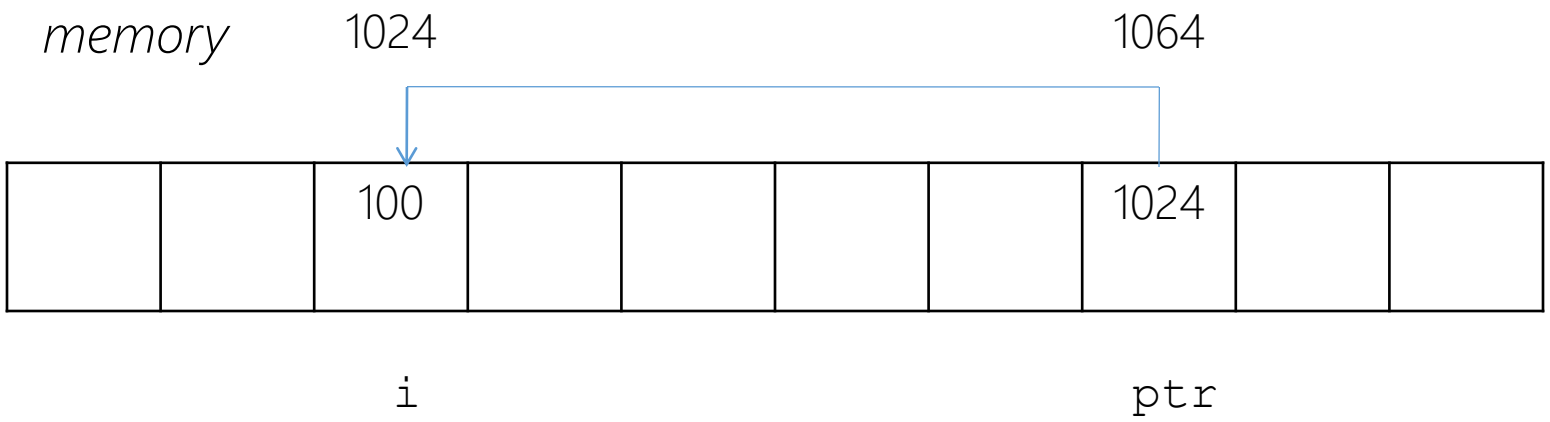
As we know, they are all the same!

```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100, * ptr;
        ptr = &i;
}
```

```
#include <iostream>
using namespace::std;
void main(void)
{
        int i=100, * ptr = &i;
}
```

Pointers

```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100 ;
        int *ptr = &i ;
}
```



Code Cracking - Pointers

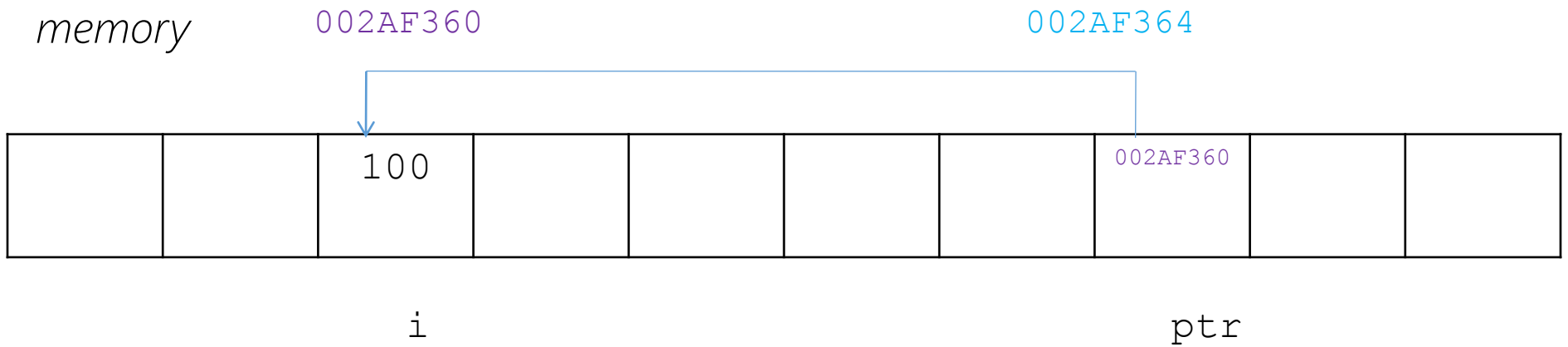
```
#include <iostream>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr;
        ptr = &i;
        cout << i << "         " << &i << endl;
        cout << ptr << "   " << &ptr << endl;
}
```

```
100         0020F180
0020F180  0020F184
Press any key to continue
```

When printing the pointer “ptr” all by itself, the output will be the address of the variable in which the pointer “ptr” points to

```
#include <iostream>
using namespace::std;
void main(void)
{
          int i = 100 ;
          int* ptr ;
          ptr = &i ;
          cout << "the address of which the pointer points to =" << ptr << endl ;
          cout << "the address of the variable =" << &i << endl;
          cout << "the address of the pointer =" << &ptr << endl ;
          cout << "the value of the variable =" << i << endl ;
          cout << "the value of the variable of which the pointer points to =" << *ptr << endl;
}
```

```
the address of which the pointer points to =002AF360
the address of the variable = 002AF360
the address of the pointer =002AF364
the value of the variable =100
the value of the variable of which the pointer points to =100
Press any key to continue . . .
```



`null` Pointers

`null` Pointers

- First, you should include <stdlib.h>
  - In VS 2008 : no need
- **`null`** is a valid address for any data type
- **`null`** is not memory address 0
- Can't dereference pointer whose value is NULL

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr;
        ptr = &i;
        if (ptr == NULL)
        {
                cout << "I'm NULL " << endl;
        }
        else
        {
                cout << "I'm not a NULL! " << endl;
        }
}
```

```
I'm not a NULL!
Press any key to continue
```

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr;
        if (ptr == NULL)
        {
                cout << "I'm NULL " << endl;
        }
        else
        {
                cout << "I'm not a NULL! " << endl;
        }
}
```

```
I'm NULL
Press any key to continue
```

The pointer hasn't been initialize yet So it's usually a NULL

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int j = 123;
        int* ptr;
        ptr = &i;
        ptr = &j;                    // dereferencing the pointer
        cout << &i << endl;
        cout << &j << endl;
        cout << ptr << endl;
}
```

```
001DF2BC
001DF2C0
001DF2C0
Press any key to continue
```

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int j = 123;
        int* ptr;
        ptr = &i;
        ptr = &j;
        cout << &i << endl;
        cout << &j << endl;
        cout << ptr << endl;
        cout << &ptr << endl;
}
```

```
0031EBBC
0031EBC0
0031EBC0
0031EBC4
Press any key to continue
```

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = &i;
        ptr = NULL;
        if (ptr == NULL)
        {
                cout << "I'm NULL " << endl;
        }
        else
        {
                cout << "I'm not a NULL! " << endl;
        }
        cout << ptr << endl;
        cout << &ptr << endl;
}
```

```
I'm NULL
00000000
001FF284
Press any key to continue
```

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr;
        cout << *ptr << endl;
}
```

```
Runtime error
```

It's a run time error to print the value of a ptr that hasn't has been initialized yet

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = &i;
        ptr = NULL;
        if (ptr == NULL)
        {
                cout << "I'm NULL " << endl;
        }
        else
        {
                cout << "I'm not a NULL! " << endl;
        }
        cout << *ptr << endl;
}
```

```
Runtime error
```

It's a runtime error to print the value of a NULL ptr

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = NULL;
        if (ptr == NULL)
        {
                cout << "I'm NULL " << endl;
        }
        else
        {
                cout << "I'm not a NULL! " << endl;
        }
}
```

```
I'm NULL
Press any key to continue
```

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = NULL;
        if (ptr = NULL)
        {
                cout << "I'm NULL " << endl;
        }
        else
        {
                cout << "I'm not a NULL! " << endl;
        }
}
```

```
I'm not a NULL!
Press any key to continue
```

```
Why?
```

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = NULL;
        if (ptr = NULL)
        {
                cout << "I'm NULL " << endl
        }
        else
        {
                cout << "I'm not a NULL! "
        }
}
```

Coz the condition is false

```
if (ptr = NULL)
```

```
I'm not a NULL!
Press any key to continue
```

Why?

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = NULL;
        if (ptr = NULL)
        {
                cout << "I'm NULL " << endl
        }
        else
        {
                cout << "I'm not a NULL! "
        }
}
```

Coz the condition is false

```
if (ptr = NULL)
if(0)
```

```
I'm not a NULL!
Press any key to continue
```

Why?

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = NULL;
        if (ptr = NULL)
        {
                cout << "I'm NULL " << endl
        }
        else
        {
                cout << "I'm not a NULL! "
        }
}
```

Coz the condition is false

```
if (ptr = NULL)
if(0)
if(false)
```

```
I'm not a NULL!
Press any key to continue
```

Why?

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
        int i = 100;
        int* ptr = NULL;
        if (ptr = NULL)
        {
                cout << "I'm NULL " << endl
        }
        else
        {
                cout << "I'm not a NULL! "
        }
}
```

Coz the condition is false

```
if (ptr = NULL)
if(0)
if(false)
Go to else
```

```
I'm not a NULL!
Press any key to continue
```

Why?

Pointers

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
          int i = 100;
          int* ptr = &i;
          *ptr = 300;
          cout << i << endl;
          cout << *ptr << endl;
}
```

```
300
300
```

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
          int i = 100;
          int* ptr =&i;
          i = 400;
          cout << i << endl;
          cout << *ptr << endl;
}
```

```
400
400
```

Pointers, Rules of Usage

Pointers

- Rules for using pointers:
  - If the pointer is initialized
    - #1: make sure the value stored in pointer is a valid address before using it
    - #2: the type is the same between the address's item and the pointer
  - If the pointer is not initialized
    - You can't use "*" then.

Pointers

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

void main(void)
{
          double i;
          double *ptr;
          ptr =&i;
}
```

Compile & run

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

void main(void)
{
          double i;
          int *ptr;
          ptr =&i;
}
```

Compiler error, the pointer is on different type from the variable it's pointing to

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

void main(void)
{
          double i;
          double *ptr;
          *ptr = 20;
}
```

Runtime error

```
void main(void)
{
          double i;
          double *ptr= &i;
          if (!ptr)
          {
          cout << "That's a bad address " << endl;
          cout << "The address is " << &ptr << endl;
          }
          else
          {
          cout << "That's a good address " << endl;
          cout << "The address is " << &ptr << endl;
          }
}
```

```
That's a good address
The address is 002DECE4
```

`!ptr` used for testing invalid address for ptr

Pointers

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

void main(void)
{
          double i;
          double *ptr= NULL;
          if (!ptr)
          {
          cout << "That's a bad address " << endl;
          cout << "The address is " << &ptr << endl;
          }
          else
          {
          cout << "That's a good address " << endl;
          cout << "The address is " << &ptr << endl;
          }
}
```

```
That's a bad address
The address is 0020F0F4
Press any key to continue
```

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

void main(void)
{
          double i;
          double *ptr;
          if (!ptr)
          {
          cout << "That's a bad address " << endl;
          cout << "The address is " << &ptr << endl;
          }
          else
          {
          cout << "That's a good address " << endl;
          cout << "The address is " << &ptr << endl;
          }

}
```

```
That's a good address
The address is 0018EF74
Press any key to continue
```

Pointers

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

void main(void)
{
          double i;
          double *ptr;
          if (!ptr)
          {
          cout << "That's a bad address " << endl;
          }
          else
          {
          cout << "That's a good address " << endl;
          cout << "The address is " << &ptr << endl;
          }

}
```

```
that's a bad address
Press any key to continue
```

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

void main(void)
{
          float x1 = 3;
          float x2 = 4;
          float *ptr1, *ptr2;
          ptr1 = &x1;
          ptr2 = &x2;

          cout << "Code_1" << endl;
          cout << x1 << " - " << *ptr1 << endl;
          cout << x2 << " - " << *ptr2 << endl;

          x1 = x1+2;
          x2 = *ptr1 * x2;

          cout << "Code_2" << endl;
          cout << x1 << " - " << *ptr1 << endl;
          cout << x2 << " - " << *ptr2 << endl;

}
```

```
Code_1
3 - 3
4 - 4
Code_2
5 - 5
20 - 20
```

Pointers

```
p1 = i1;                        // *int!= int, not correct

p1 = & i1;                      // *int = *int , correct
p2 = & i2;                      // *int = *int , correct

*p1 = i2 * 2;                   // int = int   , correct
*p2 = *p1 + 50;                 // int = int   , correct

cout << *p2 << endl;
```

Pointers

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

void main(void)
{
          int i1=100, i2=300;
          int *p1,*p2;

          p1 = & i1;// *int = *int      , correct
          p2 = & i2;// *int = *int      , correct

          *p1 = i2 * 2;   // int = int  , correct
          *p2 = *p1 + 50; // int = int  , correct

          cout << *p2 << endl;
}
```

```
650
```

Pointers

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

void main(void)
{
          char i1= 'c', i2= 's';
          char *p1,*p2;
          p2 = &i1;
          i1 = 'd';
          cout << *p2 << endl;
}
```

```
d
```

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

void main(void)
{
          char i1= 'c', i2= 's';
          char *p1,*p2;
          p2 = &i1;
          i1 = 'd';
          p1 = &i2;
          p2 = p1;
          cout << *p2 << endl;
}
```

```
s
```

Pointers

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

void main(void)
{
          char i1= 'c', i2= 's';
          char *p1,*p2;
          p2 = &i1;
          i1 = 'd';
          p1 = &i2;
          p2 = p1;
          i2 = 'k';
          cout << *p2 << endl;
          cout << *p1 << endl;
}
```

```
k
k
```

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

void main(void)
{
          char i1= 'c', i2= 'j';
          char *p1 = &i1, *p2 = &i2;
          p1 = p2;  // not the same as *p1 = *p2
          cout << p2 << endl;
          cout << p1 << endl;
          cout << *p1 << endl;
          cout << *p2 << endl;
          cout << "i1 = " << i1 << endl;
          cout << "i2 = " << i2 << endl;
}
```

```
j
j
j
j
i1 = c
i2 = j
Press any key to continue
```

Pointers

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

void main(void)
{
          char i1= 'c', i2= 'j';
          char *p1 = &i1, *p2 = &i2;
          *p1 = *p2;          // not the same as p1 = p2
          cout << p2 << endl;
          cout << p1 << endl;
          cout << *p1 << endl;
          cout << *p2 << endl;
          cout << "i1 = " << i1 << endl;
          cout << "i2 = " << i2 << endl;
}
```

```
j
j
j
j
i1 = j
i2 = j
Press any key to continue
```

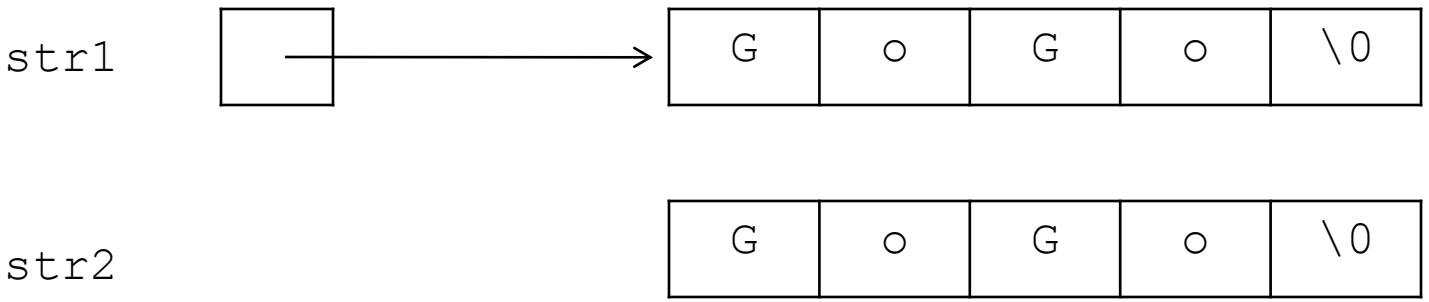
Pointers

```
p1 = p2;
```


```
*p1 = *p2;
```

Pointers

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

void main(void)
{
        char str1 [] = "GoGo";
        char *str2 = "GoGo";
}
```



Pointers

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

void main(void)
{
          char str1 [] = "GoGo";
          char *str2 = "GoGo";
          if (str1 == str2)
          {
                    cout << "The same. " << endl;

          }
          else
          {
                    cout << "Not the same. " << endl;
          }
          cout << str1 << endl;
          cout << str2 << endl;
}
```

```
Not the same.
GoGo
GoGo
Press any key to continue
```

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

void main(void)
{
          char str1 [] = "GoGo";
          char *str2 = "GoGo";
          if (str1 = str2)
          {
                    cout << "The same. " << endl;

          }
          else
          {
                    cout << "Not the same. " << endl;
          }
          cout << str1 << endl;
          cout << str2 << endl;
}
```

```
Can’t convert from char* to char[5]
```

Pointers

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

void main(void)
{
          char *str1 = "GoGo";
          char *str2 = "GoGo";
          if (str1 == str2)
          {
                    cout << "The same. " << endl;

          }
          else
          {
                    cout << "Not the same. " << endl;
          }
          cout << str1 << endl;
          cout << str2 << endl;
}
```

```
Not the same.
GoGo
GoGo
Press any key to continue
```

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

void main(void)
{
          char *str1 = "GoGo";
          char *str2 = "GoGo";
          if (str1 = str2)
          {
                    cout << "The same. " << endl;

          }
          else
          {
                    cout << "Not the same. " << endl;
          }
          cout << str1 << endl;
          cout << str2 << endl;
}
```

```
The same.
GoGo
GoGo
Press any key to continue
```

Pointers

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

void main(void)
{
          char str1[] = "GoGo";
          char str2[] = "GoGo";
          if (str1 == str2)
          {
                    cout << "The same. " << endl;

          }
          else
          {
                    cout << "Not the same. " << endl;
          }
          cout << str1 << endl;
          cout << str2 << endl;
}
```

```
Not the same.
GoGo
GoGo
Press any key to continue
```

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

void main(void)
{
          char str1[] = "GoGo";
          char str2[] = "GoGo";
          str1 = str2;
          if (str1 == str2)
          {
                    cout << "The same. " << endl;

          }
          else
          {
                    cout << "Not the same. " << endl;
          }
          cout << str1 << endl;
          cout << str2 << endl;
}
```

```
Compiler error. Assigning memory locations.
```

Pointers

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

void main(void)
{
          char str1[] = "GoGo";
          char str2[] = "GoGo";
          if (str1 = str2)
          {
                    cout << "The same. " << endl;

          }
          else
          {
                    cout << "Not the same. " << endl;
          }
          cout << str1 << endl;
          cout << str2 << endl;
}
```

Compiler error

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

void main(void)
{
          char str1[] = "GoGo";
          char str2[] = "GoGo";
          str1 = "sdsd";
          if (str1 == str2)
          {
                    cout << "The same. " << endl;

          }
          else
          {
                    cout << "Not the same. " << endl;
          }
          cout << str1 << endl;
          cout << str2 << endl;
}
```

Compiler error

References

References

- Alias
  - What does that means?
    - Creating another name for the variable, so that the Alias refers to the same memory address as does the original variable

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

void main(void)
{
        float foo;
        float &bar = foo; // must be initialized at definition time
}
```

Must be initialized at definition time

References

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

void main(void)
{
          float foo;
          float &bar;
          bar = &foo;
}
```

```
Compiler error. Must be initialized when declared
```

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

void main(void)
{
          float foo;
          float &bar = foo;
          foo = 33;
          cout << bar << endl;
          cout << foo << endl;
}
```

```
33
33
```

References

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

void main(void)
{
          float foo;
          float &bar = foo;
          cout << bar << endl;
          cout << foo << endl;
}
```

```
7.06997e+018
7.06997e+018
Press any key to continue
```

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

void main(void)
{
          float foo;
          float &bar = foo;
          float i =3;
          bar = i;
          cout << bar << endl;
          cout << foo << endl;
}
```

```
3
3
```

References

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

void main(void)
{
          float foo;
          float &bar = foo;
          float i =3;
          bar = &i;
          cout << bar << endl;
          cout << foo << endl;
}
```

Compiler error

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

void main(void)
{
          float foo;
          int &bar = foo;
          cout << bar << endl;
          cout << foo << endl;
}
```

Compiler error, types differ

Passing Parameters to Functions

- Very Important -

Passing Parameters to Functions

- By value
- By reference
  - with reference parameters
  - with pointer parameters

Passing Parameters to Functions

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

void BadSwap (int a, int b)
{
          int temp;
          temp = a;
          a = b;
          b = temp;
}


void main(void)
{
          int a = 2;
          int b = 3;
          cout << "Before Bad Swap!!" << endl;
          cout << "a is = " << a << endl;
          cout << "b is = " << b << endl;
          BadSwap(a,b);
          cout << "After Bad Swap!!" << endl;
          cout << "a becomes = " << a << endl;
          cout << "b becomes = " << b << endl;
}
```

```
Before Bad Swap!!
a is = 2
b is = 3
After Bad Swap!!
a becomes = 2
b becomes = 3
```

Passing by value

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

void Swap (int &a, int &b)
{
          int temp;
          temp = a;
          a = b;
          b = temp;
}


void main(void)
{
          int a = 2;
          int b = 3;
          cout << "Before Swap!!" << endl;
          cout << "a is = " << a << endl;
          cout << "b is = " << b << endl;
          Swap(a,b);
          cout << "After Swap!!" << endl;
          cout << "a becomes = " << a << endl;
          cout << "b becomes = " << b << endl;
}
```

```
Before Swap!!
a is = 2
b is = 3
After Swap!!
a becomes = 3
b becomes = 2
```

Passing by reference

Passing Parameters to Functions

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

void Swap (int *a, int *b)
{
          int *temp = new int;
          *temp = *a;
          *a = *b;
          *b = *temp;
}


void main(void)
{
          int a = 2;
          int b = 3;
          cout << "Before Swap!!" << endl;
          cout << "a is = " << a << endl;
          cout << "b is = " << b << endl;
          Swap(&a,&b);
          cout << "After Swap!!" << endl;
          cout << "a becomes = " << a << endl;
          cout << "b becomes = " << b << endl;
}
```

```
Before Swap!!
a is = 2
b is = 3
After Swap!!
a becomes = 3
b becomes = 2
```

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

void Swap (int *a, int *b)
{
          int *temp = new int;
          *temp = *a;
          *a = *b;
          *b = *temp;
}


void main(void)
{
          int a = 2;
          int b = 3;
          cout << "Before Bad Swap!!" << endl;
          cout << "a is = " << a << endl;
          cout << "b is = " << b << endl;
          Swap(a,b);
          cout << "After Bad Swap!!" << endl;
          cout << "a becomes = " << a << endl;
          cout << "b becomes = " << b << endl;
}
```

```
Compiler error
Can't convert
int to *int
```

Passing Parameters to Functions

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

void Swap (int *a, int *b)
{
          int *temp;
          *temp = *a;
          *a = *b;
          *b = *temp;
}


void main(void)
{
          int a = 2;
          int b = 3;
          cout << "Before Bad Swap!!" << endl;
          cout << "a is = " << a << endl;
          cout << "b is = " << b << endl;
          Swap(&a,&b);
          cout << "After Bad Swap!!" << endl;
          cout << "a becomes = " << a << endl;
          cout << "b becomes = " << b << endl;
}
```

```
Runtime error
temp hasn't been initialized first!
```

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

void Swap (int *a, int *b)
{
          int *temp = new int;
          temp = a;
          a = b;
          b = temp;
}


void main(void)
{
          int a = 2;
          int b = 3;
          cout << "Before Bad Swap!!" << endl;
          cout << "a is = " << a << endl;
          cout << "b is = " << b << endl;
          Swap(&a,&b);
          cout << "After Bad Swap!!" << endl;
          cout << "a becomes = " << a << endl;
          cout << "b becomes = " << b << endl;
}
```

```
Before Bad Swap!!
a is = 2
b is = 3
After Bad Swap!!
a becomes = 2
b becomes = 3
```

Coz the pointers are returned to their original pointing locations after the method is done!

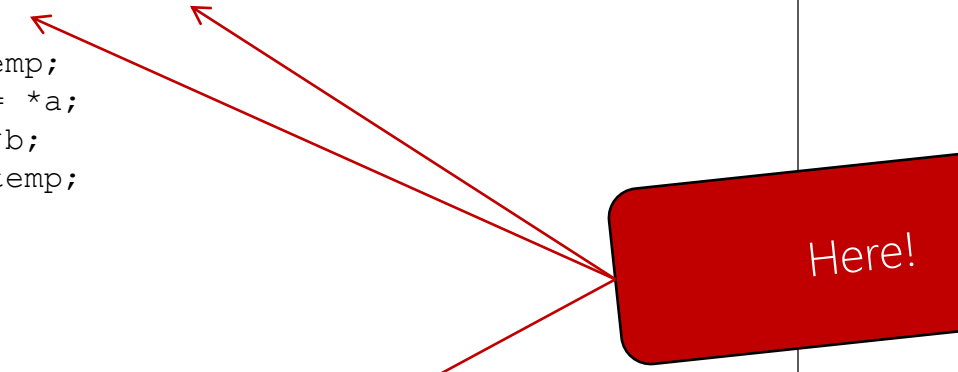
Passing Parameters to Functions

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

void Swap (int *a, int *b)
{
          int temp;
          temp = *a;
          *a = *b;
          *b = temp;
}


void main(void)
{
          int *a = new int;
          int *b = new int;
          *a = 3;
          *b = 5;
          cout << "Before Swap!!" << endl;
          cout << "a is = " << *a << endl;
          cout << "b is = " << *b << endl;
          Swap(a,b);
          cout << "After Swap!!" << endl;
          cout << "a becomes = " << *a << endl;
          cout << "b becomes = " << *b << endl;
}
```

```
Before Swap!!
a is = 3
b is = 5
After Swap!!
a becomes = 5
b becomes = 3
```

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

void Swap (int *a, int *b)
{
          int temp;
          temp = *a;
          *a = *b;
          *b = temp;
}


void main(void)
{
          int *a = new int;
          int *b = new int;
          *a = 3;
          *b = 5;
          Swap(&a,&b);
          cout << *a << endl;
          cout << *b << endl;
}
```

```
Compiler error Can't
convert **int to *int
```



Passing Parameters to Functions

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

void Swap (int *a, int *b)
{
          int temp;
          temp = *a;
          *a = *b;
          *b = temp;
}


void main(void)
{
          int *a;
          int *b;
          *a = 3;
          *b = 5;
          Swap(a,b);
          cout << *a << endl;
          cout << *b << endl;
}
```

Compile but runtimer error. The is because the pointers hasn't been initialized (without new)

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

void Swap (int *a, int *b)
{
          int temp;
          temp = *a;
          a = b;
          *b = temp;
}


void main(void)
{
          int *a = new int;
          int *b = new int;
          *a = 3;
          *b = 5;
          Swap(a,b);
          cout << *a << endl;
          cout << *b << endl;
}
```

```
3
3
```

Passing Parameters to Functions

```
#include <iostream>
using namespace::std;
int* f ()
{
        int c = 2;
        cout << "the address of c in f fuction " << &c << endl;
        return (&c);
}

void Wrong()
{
        int* p;
        cout << "The address of the p before calling f is = " << p << endl;
        p = f();
        cout << "The address of the p after calling f is = " << p << endl;
        cout << "The value of the p is now!!! = " << *p << endl;
}

void main(void)
{Wrong();}
```

```
The address of the p before calling f is = 00000000
the address of c in f fuction 0014F190
The address of the p after calling f is = 0014F190
The value of the p is now!!! = 1372776
Press any key to continue
```

Function must not return a pointer to a local variable in the function

Passing Parameters to Functions

```
#include <iostream>
using namespace::std;
int f ()
{
        int c = 2;
        cout << "the address of c in f fuction " << &c << endl;
        return (c);
}

void Wrong()
{
        int* p = new int;
        cout << "The address of the p before calling f is = " << p << endl;
        *p = f();
        cout << "The address of the p after calling f is = " << p << endl;
        cout << "The value of the p is now!!! = " << *p << endl;
}
void main(void)
{Wrong();}
```

```
The address of the p before calling f is = 00311DE0
the address of c in f fuction 0028EEF8
The address of the p after calling f is = 00311DE0
The value of the p is now!!! = 2
Press any key to continue
```

Passing Parameters to Functions

```
#include <iostream>
using namespace::std;
int f ()
{
        int c = 2;
        cout << "the address of c in f fuction " << &c << endl;
        return (c);
}

void Wrong()
{
        int* p;
        cout << "The address of the p before calling f is = " << p << endl;
        *p = f();
        cout << "The address of the p after calling f is = " << p << endl;
        cout << "The value of the p is now!!! = " << *p << endl;
}
void main(void)
{Wrong();}
```

```
Runtime error. No memory allocation (missing new)
```

Passing Parameters to Functions

```
void swap1 (int x, int y)
{
     int temp = x;
      x = y;
     y = temp;}

void swap2 (int *x, int *y)
{
     int temp = *x;
     *x = *y;
     *y = temp;}

void swap3 (int &x, int &y )
{
     int temp = x;
      x = y;
     y = temp; }

void main(void)
{
          int i1 = 5;
          int i2 = 7;
//How should we call
//the function
}
```

```
void main(void)
{
          int i1 = 5;
          int i2 = 7;
          swap1(i1,i2);
          swap2(&i1,&i2);
          swap3(i1,i2);
}
```

```
#include <iostream>
using namespace::std;
void swap1 (int x, int y){
int temp = x;
x = y;
y = temp;}
void swap2 (int *x, int *y){
int temp = *x;
*x = *y;
*y = temp;}

void swap3 (int &x, int &y )
{
int temp = x; x = y;y = temp; }

void main(void)
{int i1 = 5, i2 = 7;
swap1(i1,i2);
cout << "After 1st sawp function " << endl;
cout << "i1 = " << i1 <<endl;
cout << "i2 = " << i2 << endl;
swap2(&i1,&i2);
cout << "After 2nd sawp function " << endl;
cout << "i1 = " <<i1 <<endl;
cout << "i2 = " << i2 << endl;
swap3(i1,i2);
cout << "After 3rd sawp function " << endl;
cout << "i1 = " << i1 <<endl;
cout << "i2 = " << i2 << endl;                    }
```

```
After 1st sawp function
i1 = 5
i2 = 7
After 2nd sawp function
i1 = 7
i2 = 5
After 3rd sawp function
i1 = 5
i2 = 7
```

Dynamic Memory Allocation

Dynamic Memory Allocation

- What Dynamic means?
- What Memory Allocation means?
- What Dynamic Memory Allocation means?

Dynamic Memory Allocation

- When Dynamic Memory Allocation is used?
    - Array that its extent is not known till Runtime
    - Objects that needs to be created but not known till Runtime

- What Dynamic Memory Allocation do?
  - Allocate and de-allocate memory at run time depending on the information at running time

- new \ delete keywords (Dynamic)

Dynamic Memory Allocation

- Static Memory Allocation
  - Allocated at compile time
- Dynamic Memory Allocation
  - Allocated at Run time

Dynamic Memory Allocation
Pointers and Arrays

Pointers and Arrays

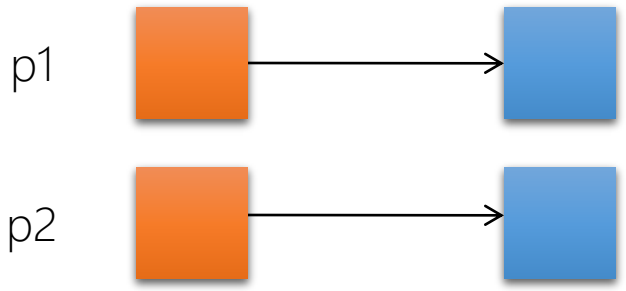
- The name of "Array" is the address of the its first element in memory

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


void main(void)
{
        int a[200];     // Static, Allocate at compile time

}
```

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


void main(void)
{
        int ArrLength;
        int *ptr;
        cin >> ArrLength;
        ptr = new int[ArrLength];      // Dynamic, allocate at Runtime

}
```

Pointers and Arrays

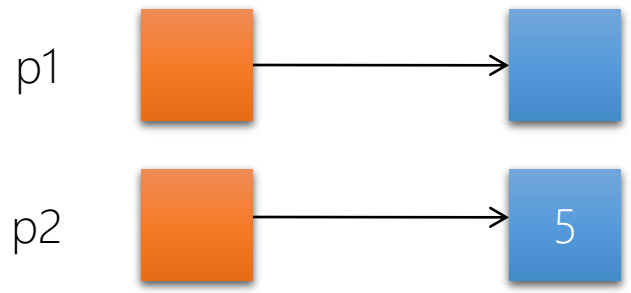
```
#include <iostream>
using namespace::std;


void main(void)
{
        int *p1, *p2;
        p1 = new int;  // allocate but not initialized yet
        p2 = new int; // allocate but not initialized yet
}
```




```
-842150451
546546546
```

Pointers and Arrays

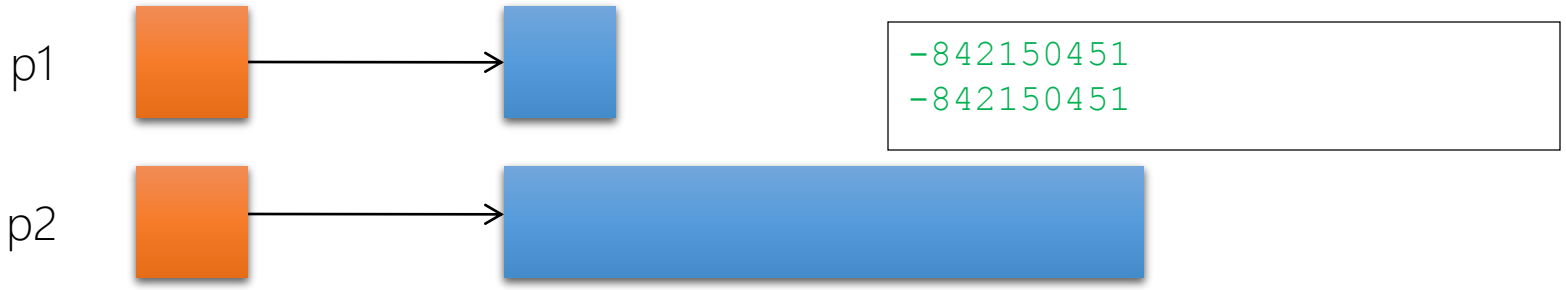
```
#include <iostream>
using namespace::std;


void main(void)
{
        int *p1, *p2;
        p1 = new int;              // allocate but not initialized yet
        p2 = new int (5);          // allocate and initialized
        cout << *p1 << endl;
        cout << *p2 << endl;
}
```




```
-842150451
5
```

Pointers and Arrays

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


void main(void)
{
        int *p1, *p2;
        p1 = new int;
        p2 = new int [5];
        cout << *p1 << endl;
        cout << *p2 << endl;
}
```




```
-842150451
-842150451
```

Pointers and Arrays

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


void main(void)
{
          int *p1;
          p1 = new int [5];
          *p1 = 2;
          *(p1+1) = 3;
          cout << *p1 << endl;
          cout << *(p1+1) << endl;
}
```

```
2
3
```

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


void main(void)
{
          int *p1;
          p1 = new int [5];
          *p1 = 2;
          *(p1+1) = 3;
          cout << *p1 << endl;
          cout << *(p1+1) << endl;
          cout << *(p1+2) << endl;
}
```

```
2
3
-842150451
Press any key to continue
```

Pointers and Arrays

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


void main(void)
{
          int *p1;
          p1 = new int [5];
          *p1 = 2;
          *(p1+1) = 4;
          cout << *p1 << endl;
          cout << *(p1+1) << endl;
          cout << *p1+1 << endl;
          // again, the array name is the address
          // of its first element in memory
}
```

```
2
4
3
Press any key to continue
```

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


void main(void)
{
          int *p1;
          p1 = new int [5];
          *p1 = 2;
          *p1+1 = 4;
}
```

```
Compiler error
```

Pointers and Arrays

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


void main(void)
{
          int *p1;
          p1 = new int [5];
          *p1 = 2;
          *(p1++) = 4;
          cout << *p1 << endl;
          cout << *(p1+1) << endl;
          cout << *(p1-1) << endl;

}
```

```
-842150451
-842150451
4
Press any key to continue
```

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


void main(void)
{
          int *p1;
          p1 = new int [5];
          *p1 = 2;
          cout << *p1 << endl;
          cout << p1 << endl;
          delete p1;
          cout << *p1 << endl;
          cout << p1 << endl;

}
```

```
2
006D1848
-572662307
006D1848
Press any key to continue
```

Pointers and Arrays

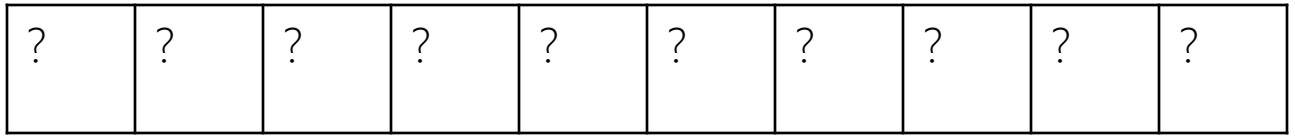
```
#include <iostream>
using namespace::std;


void main(void)
{
          int *p1;
          p1 = new int [5];
          *p1 = 2;
          cout << *p1 << endl;
          cout << p1 << endl;
          delete p1;
          cout << *p1 << endl;
          cout << p1 << endl;
          *p1 = 3;
          // Re-allocating without new!!!!!, it works!!!
          cout << *p1 << endl;
          cout << p1 << endl;
}
```

```
2
00061848
-572662307
00061848
3
00061848
```

Pointers and Arrays

- Now, Memory is allocated
  - But not initialized

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


void main(void)
{
        int Arr[10];
}
```



Pointers and Arrays

- Now, Memory is allocated
  - But not initialized

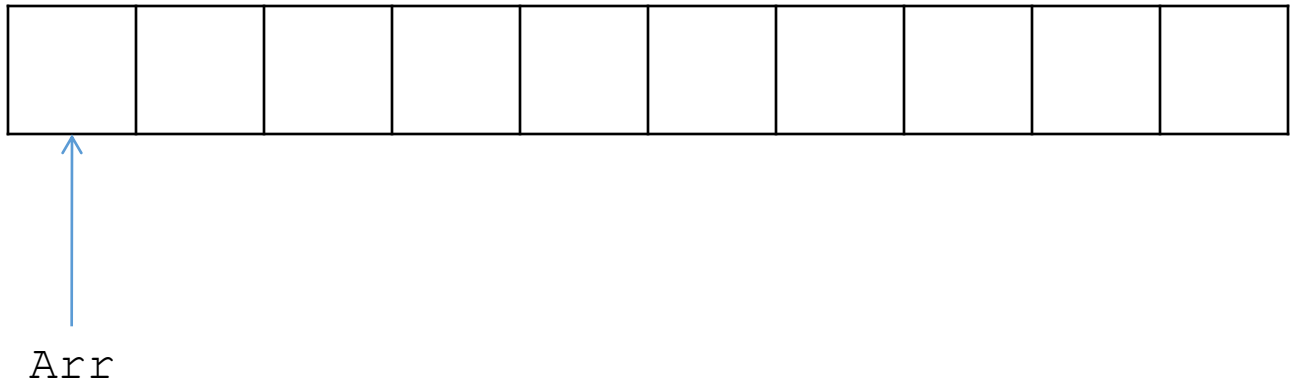
```
#include <iostream>
using namespace::std;


void main(void)
{
        int Arr[10];
        Arr[9] = 2;
        cout << Arr[9] << endl;
        cout << Arr[2] << endl;
}
```

?	?	?	?	?	?	?	?	?	2

Pointers and Arrays

- In C++, the array is really just a pointer to its first element

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


void main(void)
{
        int Arr[10];
}
```

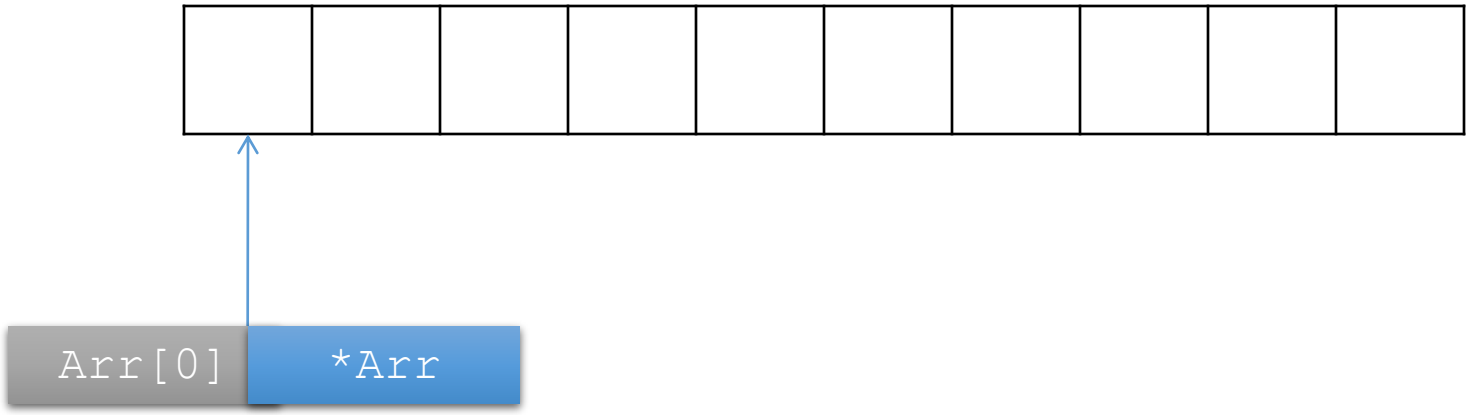


Pointers and Arrays

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


void main(void)
{
          int Arr[8]= {1,2,4,78,9,2,7,3};
          if (Arr == &Arr[0])
          {
                    cout << "This will ALWAYS be true!" << endl;
          }
}
```

```
This will ALWAYS be true!
Press any key to continue
```

Pointers and Arrays




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


void main(void)
{
        int Arr[8];
        cout << Arr[0] << endl;
        cout << *Arr << endl;
        cout << *(&Arr[0]) << endl;
}
```

```
1306992
1306992
1306992
```

Pointers and Arrays

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


void main(void)
{
          int Arr[] = {1,2,3};
          int *ArrPtr = Arr;
          ArrPtr++;
          ArrPtr++;
          ArrPtr-=2;
          cout << *(ArrPtr++) << endl;
          ArrPtr--;
          ArrPtr = Arr;
          ArrPtr += 2;
          cout << ArrPtr - Arr << endl;
          cout << *ArrPtr << endl;
          cout << *Arr << endl;

}
```

```
1
2
3
1
```

Pointers and Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int sum = 0;
          int Arr[ArrSize] = {1,3};
          for (int i=0; i<ArrSize; i++)
          {
                    sum += Arr[i];
          }
          cout << "The sum is = " <<  sum << endl;
}
```

```
4
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int sum = 0;
          int Arr[ArrSize] = {1,3};
          int *ptr;
          for (ptr = Arr; ptr < Arr+ArrSize-1; ++ptr)
          {
                    sum += *ptr;
          }
          cout << "The sum is = " <<  sum << endl;
}
```

```
4
```

Pointers and Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int sum = 0;
          int Arr[ArrSize] = {1,3};
          int *ptr;
          for (ptr = Arr; ptr < &Arr[ArrSize]; ++ptr)
          {
                    sum += *ptr;
          }
          cout << "The sum is = " <<  sum << endl;
}
```

```
4
```

Pointers and Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int sum = 0;
          int Arr[ArrSize] = {1,3};
          int* ptr = Arr;
          int i = 2;
          cout << ptr[1] << endl;
          cout << ptr[2] << endl;
}
```

```
3
0
Press any key to continue
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int Arr[10];
          for (int i = 0; i< 10; i++)
          {
                    Arr[i] = i;
          }
          int* p = Arr;
          cout << *p << endl;
          cout << *p+1 << endl;
          cout << *p++ << endl;
          cout << *++p << endl;
          cout << *(p+1) << endl;
          cout << *(p+5) << endl;
}
```

```
0
1
0
2
3
7
```

Pointers and Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int Arr[10];
          int *Iter = &Arr[0];
          int *Iter_End = &Arr[10];
          int i = 0;
          while (Iter!= Iter_End)
          {
          cout << "We're in elemet #" << i
          << ", with value = " << *Iter << endl;
                    ++Iter;
                    i++;
          }
}
```

```
We're in elemet #0, with value = 2429382
We're in elemet #1, with value = 1371168
We're in elemet #2, with value = 1518572499
We're in elemet #3, with value = 1
We're in elemet #4, with value = 3336456
We're in elemet #5, with value = 1371200
We're in elemet #6, with value = 2430657
We're in elemet #7, with value = 1371888
We're in elemet #8, with value = 1371336
We're in elemet #9, with value = 3336456
```

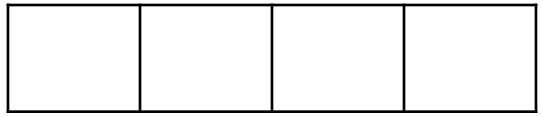
```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int Arr[10] = {};
          int *Iter = &Arr[0];
          int *Iter_End = &Arr[10];
          int i = 0;
          while (Iter!= Iter_End)
          {
          cout << "We're in elemet #" << i
          << ", with value = " << *Iter << endl;
                    ++Iter;
                    i++;
          }
}
```

```
We're in elemet #0, with value = 0
We're in elemet #1, with value = 0
We're in elemet #2, with value = 0
We're in elemet #3, with value = 0
We're in elemet #4, with value = 0
We're in elemet #5, with value = 0
We're in elemet #6, with value = 0
We're in elemet #7, with value = 0
We're in elemet #8, with value = 0
We're in elemet #9, with value = 0
```

Pointers and Arrays

- Dynamic arrays
  - Size is not specified at compile time but at Run time!
    - By using new

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
        int *ptr = new int [4]; // allocating 4 interger elements
}
```



Pointers and Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int *ptr = new int [];
          cout << *ptr << endl;
          cout << *(ptr+3) << endl;
}
```

```
-33686019
-1962934272
Press any key to continue
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int *ptr = new int [];
          cout << *ptr << endl;
          cout << *(ptr+3) << endl;
          delete [] ptr;
          cout << *ptr << endl;
}
```

```
-33686019
-1962934272
-572662307
Press any key to continue
```

Pointers and Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int *ptr = new int [6];
          cout << *ptr << endl;
          cout << *(ptr+3) << endl;
          delete [] ptr;
          cout << *ptr << endl;
}
```

```
-842150451
-842150451
-572662307
Press any key to continue
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int *ptr = new int [];
          cout << *ptr << endl;
          cout << *(ptr+3) << endl;
          delete ptr [];
          cout << *ptr << endl;
}
```

```
Compiler error The syntax for deleting is delete []
ptr;
```

Pointers and Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          float * pArr;
          int Length;
          cin >> Length;       // 5
          pArr = new float [Length];
          for (int i= 0; i < Length; i++ )
          {
                    pArr[i] = i;
          }

          cout << "The array is: " << endl;
          for (int i= 0; i < Length; i++ )
          {
                    cout << pArr[i] << endl;
          }
}
```

```
5
The array is:
0
1
2
3
4
```

```
0 - 0x000441C40
2 - 0x000441C48
3 - 0x000441C50
4 - 0x000441C58
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          float * pArr;
          int Length;
          cin >> Length;       // 5
          pArr = new float [Length];
          for (float i= 0; i < Length; i++ )
          {
                    pArr[i] = i;
          }

          cout << "The array is: " << endl;
          for (int i= 0; i < Length; i++ )
          {
                    cout << pArr[i] << endl;
          }
}
```

```
Compiler error
Counter must be an integral type
```

Pointers and Strings

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char KoKo[] = "Hello!"; // defining a string
          char *Ptr = KoKo;
          *Ptr = 'C';
          *Ptr++ = 'V';
          *Ptr = 'B';
          cout << KoKo << endl;
}
```

```
VBllo!
Press any key to continue
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char KoKo[] = "Hello!"; // defining a string
          char *Ptr = KoKo;
          *Ptr = 'C';
          *Ptr++ = "V";
          *Ptr = 'B';
          cout << KoKo << endl;
}
```

```
Compiler error,   ""
```

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char *p = new char [10];
          cout << *(p+9) << endl;

}
```

```
=
```

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char *p = new char [10];
          cin >> p;
          cout << p << endl;
          cout << *(p+9) << endl;

}
```

```
123456789
123456789

Press any key to continue
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char *p = new char [10];
          cin >> p;
          cout << p << endl;
          cout << *(p+9) << endl;

}
```

```
32423
32423
=
Press any key to continue
```

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char *p = new char [10];
          cin >> p;
          cout << p << endl;
          cout << *(p+9) << endl;

}
```

```
123456789123456789
123456789123456789
1
Press any key to continue
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char *p = new char [10];
          cin >> p;
          p[9]='\0';          // now used to cut down
the string
                              // heeeeeehaaaaaaa!!!:D
          cout << p << endl;
          cout << *(p+9) << endl;

}
```

```
213124234324234
213124234

Press any key to continue
```

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char *p = new char [10];
          cin >> p;
          p[9]='\0';// now used to cut down the string
                    // heeeeeehaaaaaaa!!!:D
          cout << p << endl;
          cout << *(p+9) << endl;

}
```

```
234234
234234

Press any key to continue
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char *p = "COOL MAN!";
          cout << p << endl;
          cout << *p << endl;
          cout << &*p << endl;
          cout << *&p << endl;
          cout << *(p+3) << endl;

}
```

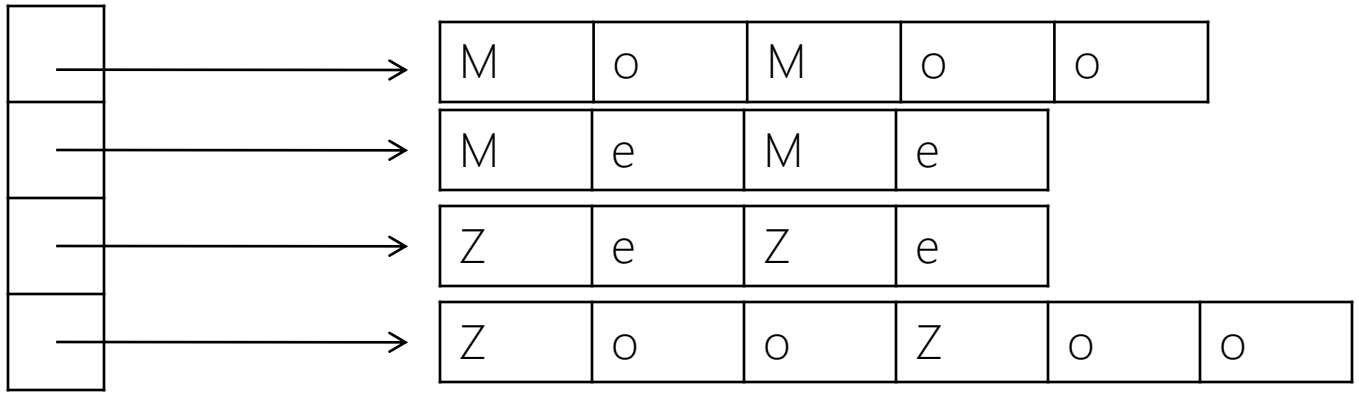
```
COOL MAN!
C
COOL MAN!
COOL MAN!
L
```

Pointers and Multi-dimensional Arrays

Pointers and Multi-dimensional Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
            char * WoW[4] = {"MoMoo", "MeMe", "ZeZe", "ZooZoo"};
}
```

What does that means?!



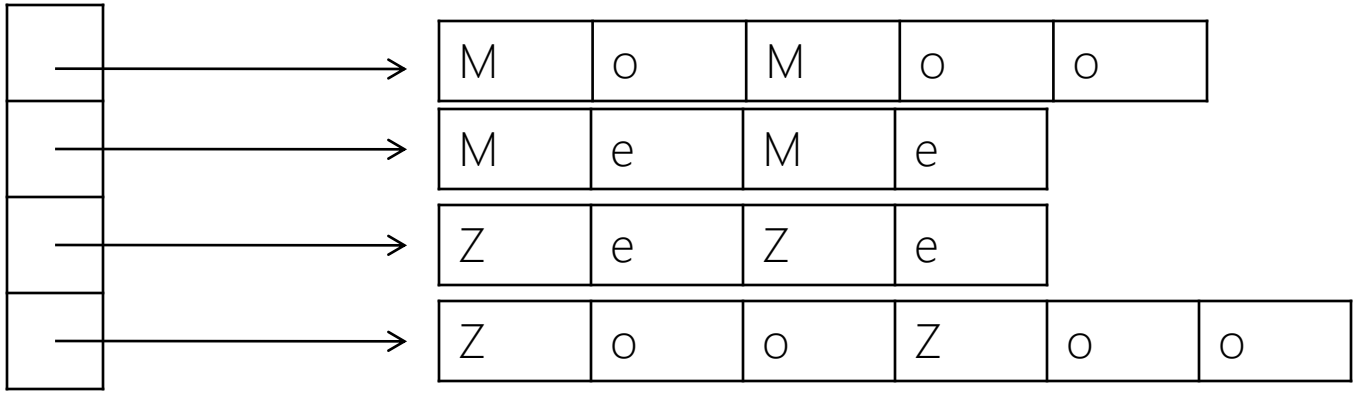
Pointers and Multi-dimensional Arrays

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
            char * WoW[4] = {"MoMoo", "MeMe", "ZeZe", "ZooZoo"};
}
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
            char * WoW[] = {"MoMoo","MeMe", "ZeZe", "ZooZoo"};
}
```

Size of the arrays is determined at compile time

Both are the same

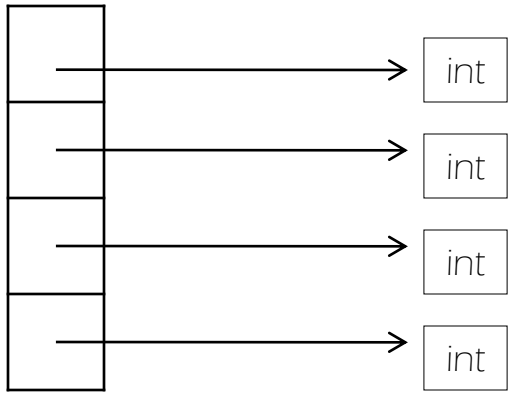


Pointers and Multi-dimensional Arrays

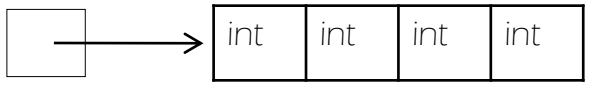
What's the difference?!

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
            int *ThatIsCrazy1 [4];
            int (*ThatIsCrazy2) [4];
}
```

ThatIsCrazy1



ThatIsCrazy2



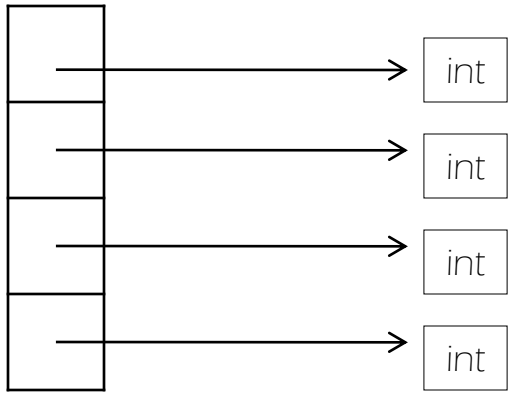
Pointers and Multi-dimensional Arrays

What's the difference?!

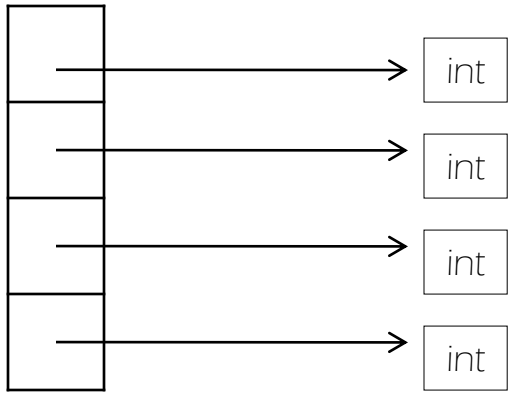
```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
            int *ThatIsCrazy1 [4];
            int (*ThatIsCrazy2) [4];
}
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
            int *ThatIsCrazy1 [4];
            int *(ThatIsCrazy2 [4]);
}
```

ThatIsCrazy1



ThatIsCrazy2



const Pointers

const Pointers

- What does const for pointers means?
  - ptr is a pointer which points to a const integer?
  - OR, ptr is a pointer (not a const pointer)?

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
        const int *ptr;
}
```

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i = 2;
          const int *ptr;
          ptr = &i; // changing the pointer
}
```

Compile and run

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i = 2;
          const int *ptr;
          ptr = &i;
          *ptr = 3;
}
```

Compiler error Coz the thing which the pointer points to is const and shouldn’t be modified!

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i = 2;
          const int *ptr;
          ptr = &i;
          cout << *ptr << endl;
}
```

2

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i;
          int *const ptr =&i;
}
```

What does this means?

Ptr is a pointer which is const which points to integer

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i;
          int *const ptr;
}
```

Compiler error

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i,j;
          int *const ptr =&i;
          ptr = &j;
}
```

Compiler error

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i=2, j;
          int *const ptr =&i;
          cout << *ptr << endl;
}
```

```
2
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i, j;
          int *const ptr =&i;
          cout << *ptr << endl;
}
```

```
2458743545
```

const Pointers

- Pointer to const
  - const int *ptr;
  - int const *ptr;

- const pointer
  - int *const ptr;

const Pointers

- Const pointer to const

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
        int i = 3;
        const int *ptr1;                // p
        int const *ptr2;                // p
        int *const ptr3 = &i;           // c
        const int *const ptr4 = &i;     // c
        int const *const ptr5 = &i;     // c

}
```

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i = 3;
          const int *ptr1;              // pointer to
const
          int const *ptr2;              // pointer to
const
          int *const ptr3;              // const pointer
          const int *const ptr4;        // const pointer
to const
          int const *const ptr5;        // const pointer
to const

}
```

```
Compiler error for:
ptr3
ptr4
ptr5
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i = 3, j = 4;
          const int *const ptr1 = &i;   // const pointer
to const
          int const *const ptr3 = &i;   // const pointer
to const
          ptr1 = &j; // attempting to dereferance and
change pointer

}
```

```
Compiler error
```

Pointers and Strings

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i = 3, j = 4;
          // const pointer to const
          const int *const ptr1 = &i;

          // const pointer to const
          int const *const ptr3 = &i;

          *ptr1 = 3; // change the int
}
```

Compiler error

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int i = 3, j = 4;
          // const pointer to const
          const int *const ptr1 = &i;

          // const pointer to const
          int const *const ptr3 = &i;

          // Attempting to dereferance
          // and change the pointer
          ptr1 = &j;
}
```

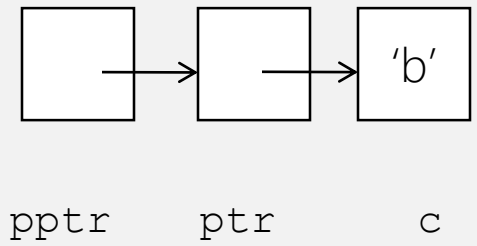
Compiler error

Pointers to Pointers

Pointers to Pointers

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char c = 'c';
          char *ptr;
          char **pptr;
          c = 'b';
          ptr = &c;
          pptr = &ptr;
          cout << c << endl;
          cout << *ptr << endl;
          cout << ptr << endl;
          cout << &c << endl;
          cout << pptr << endl;
          cout << &ptr << endl;
          cout << &pptr << endl;
}
```

```
b
b
b
b
0028EF54
0028EF54
0028EF4C
```



`void` Pointers

void Pointers

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{          int *iptr;
           void *vptr;
           int i = 3;
           float f = 4;

           iptr = &i;
           //iptr = &f; // error

           vptr = &i;
           cout << *vptr << endl;
           cout << &vptr << endl;
           cout << vptr << endl;

           vptr = &f;
           cout << *vptr << endl;
           cout << &vptr << endl;
           cout << vptr << endl;
}
```

```
Compiler error. Why?
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
           int *iptr;
           void *vptr;
           int i = 3;
           float f = 4;

           iptr = &i;
           vptr = &i;

           cout << (*((int*)vptr)) << endl;
           cout << &vptr << endl;
           cout << vptr << endl;
           vptr = &f;
           cout << (*((float*)vptr)) << endl;
           cout << &vptr << endl;
           cout << vptr << endl;

}
```

```
3
0016EEC4
0016EEB8
4
0016EEC4
0016EEBC
```

Returning Pointers

Returning Pointers

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


int[] GoToFun(int A[2])
{
          cout << A[0] << endl;
          A[0] = 1;
          return A;
}

void main(void)
{
          int Arr[] = {3,4};
          GoToFun(Arr);
}
```

```
Compiler error. Arrays can't be used as a return type
of functions. So, how can we do it?!?!!!
```

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


int* GoToFun(int *p1)
{
          cout << p1[0] << endl;
          p1[0] = 1;
          cout << p1[0] << endl;
          return p1;
}

void main(void)
{
          int Arr[] = {3,4};
          int *ptr = Arr;
          cout << "In main, " << *ptr << endl;
          GoToFun(ptr);
          cout << "In main, " << *ptr << endl;
}
```

```
In main, 3
3
1
In main, 1
```

Returning Pointers

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


int* GoToFun(int *p1)
{
          cout << p1[0] << endl;
          p1[0] = 1;
          cout << p1[0] << endl;
          return p1;
}

void main(void)
{
          int Arr[] = {3,4};
          int *ptr = &Arr;
          cout << "In main, " << *ptr << endl;
          GoToFun(ptr);
          cout << "In main, " << *ptr << endl;
}
```

```
Error     1         error C2440: 'initializing': cannot
convert from 'int (*)[2]' to 'int *'
          c:\Users\ZGTR\Documents\Visual Studio
2008\Projects\TempC++File++\TempC++File++\MyFile.cpp
          16        TempC++File++
```

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


int& Reproducer(int A[], int index)
{
          return A[index];
}

void main(void)
{
          int i = 5;
          int Arr[] = {3,4,7};
          cout << "Before" << endl;
          for (i = 0; i < 3;++i)
          {
                    cout << Arr[i] << endl;;;;;
          }
          Reproducer(Arr,1) = -10;
          cout << "After" << endl;
          for (i = 0; i < 3;++i)
          {
                    cout << Arr[i] << endl;;;;;
          }

}
```

```
Before
3
4
7
After
3
-10
7
```

Returning Pointers

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


int& Reproducer(int A[], int index)
{
          return &A[index];
}

void main(void)
{
          int i = 5;
          int Arr[] = {3,4,7};
          cout << "Before" << endl;
          for (i = 0; i < 3;++i)
          {
                    cout << Arr[i] << endl;;;;;
          }
          Reproducer(Arr,1) = -10;
          cout << "After" << endl;
          for (i = 0; i < 3;++i)
          {
                    cout << Arr[i] << endl;;;;;
          }

}
```

```
Compiler error. &A[index];
```

Memory Problems

Memory Problems

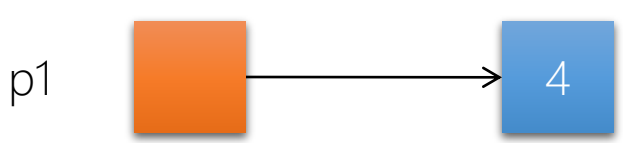
- Memory leak
  - Loss of available memory space
  - Occurs when the dynamic memory space has been allocated but never de-allocated

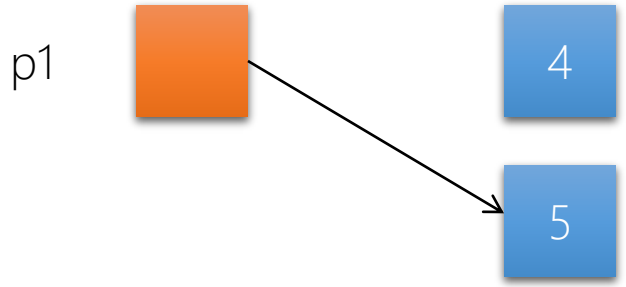
- Dangling pointer
  - A pointer that points to dynamic memory that has been de-allocated

Memory Problems - LEAK

- Now 4 is allocated but can never been reached or used

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


void main(void)
{
        int *p;
        p = new int;
        *p = 4;
        p = new int;
        *p = 5;
}
```



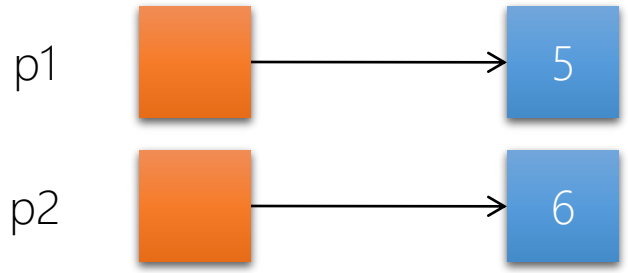


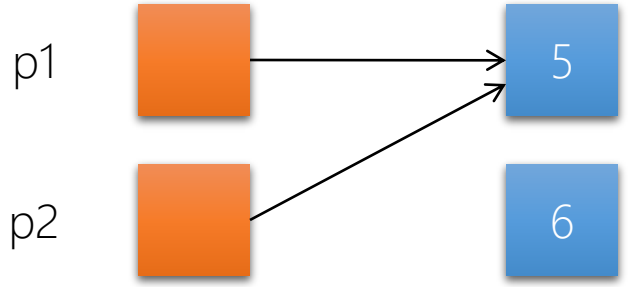
Memory Problems - LEAK

- Now 6 is

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


void main(void)
{
        int *p1;
        p1 = new int (5);
        *p1 = 4;
        int *p2 = new int (6);
        p2 = p1;
        cout << p1 << " - " << *p1 << endl;
        cout << "p1 address " << &p1 << endl;
        cout << p2 << " - " << *p2 << endl;
        cout << "p2 address " << &p2 << endl;
}
```

```
00401DE0 - 4
p1 address 0028EEA4
00401DE0 - 4
p2 address 0028EEA0
```

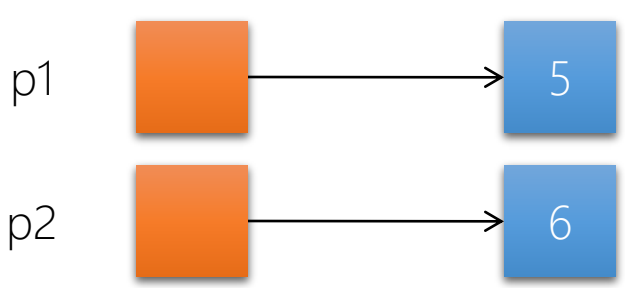


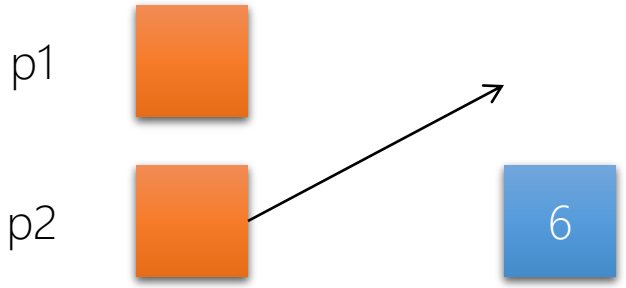


Memory Problems - Dangling

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


void main(void)
{
        int *p1;
        p1 = new int (5);
        int *p2 = new int (6);
        p2 = p1;
        delete p1;
        cout << p1 << " - " << *p1 << endl;
        cout << "p1 address " << &p1 << endl;
        cout << p2 << " - " << *p2 << endl;
        cout << "p2 address " << &p2 << endl;
}
```

```
00271DE0 - -33686272
p1 address 0025F064
00271DE0 - -33686272
p2 address 0025F060
```





Code Cracking

Code Cracking

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


void main(void)
{
          int *p;
          if(1)
          {
                    int x= 343299;
                    *p = x;
          }
          cout << *p << endl;
}
```

```
Compile but runtime error
```

```
#include <iostream>
using namespace::std;
int *p;

void SthWrong()
{
          int b = 3;
          p = &b;
}


void main(void)
{
          cout << p << endl;
          cout << &p << endl;
          SthWrong();
          cout << p << endl;
          cout << &p << endl;
          cout << *p << endl;
}
```

```
00000000
00300974
0019EE68
00300974
1699624
```

Code Cracking

```
#include <iostream>
using namespace::std;
int *p;

void SthWrong()
{
          int b = 3;
          *p = b;
}


void main(void)
{
          cout << p << endl;
          cout << &p << endl;
          SthWrong();
          cout << p << endl;
          cout << &p << endl;
          cout << *p << endl;
}
```

```
Compile but runtime error
Missing new
```

```
#include <iostream>
using namespace::std;
int *p;

void SthWrong()
{
          int b = 3;
          *p = b;
}


void main(void)
{
          p = new int;
          cout << p << endl;
          cout << &p << endl;
          SthWrong();
          cout << p << endl;
          cout << &p << endl;
          cout << *p << endl;
}
```

```
005B1DE0
00310978
005B1DE0
00310978
3
```

Code Cracking

```
#include <iostream>
#include <stdlib.h>
using namespace::std;
void main(void)
{
          int i = 100;
          int* ptr = NULL;
          if (ptr = NULL)
          {
                    cout << "I'm NULL " << endl;
          }
          else
          {
                    cout << "I'm not a NULL! " << endl;
          }
}
```

```
I'm not a NULL!
Press any key to continue
```

```
#include <iostream>
using namespace::std;
void main(void)
{
          int i = 100;
          int *ptr =&i;
          *ptr = int(&i);
          cout << *ptr << endl;
          cout << &i << endl;
}
```

```
2223152
0021EC30
Press any key to continue
```

Code Cracking

```
#include <iostream>
using namespace::std;
void main(void)
{
          int i = 100;
          int *ptr;
          *ptr = &i;
}
```

```
Compiler error, can't convert from *int to int
```

Ever dreamed to pass
a function as a parameter?

Function Pointers

Function Pointers

Delegates, Event Handling, etc.

Function Pointers

Allows operations with pointers to functions

Function pointers contains the address of the function in memory

Function Pointers

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

int Add(int a, int b ){       return a+b; }
int Sub(int a, int b ){       return a-b; }

int (*AddPtr) (int,int) = Add;
int (*SubPtr) (int,int) = Sub;

int OperationToDo (int x, int y, int (*func) (int,int))
{         int Result;
          Result = (*func)(x,y);
          return Result;}

void main(void)
{         int a=3, b=4;
          int Result;
          Result = OperationToDo(a,b,AddPtr);
          cout << "a=" << a << endl;
          cout << "b=" << b << endl;
          cout << "Result =" << Result << endl;

          Result = OperationToDo(b,a,SubPtr);
          cout << "a=" << a << endl;
          cout << "b=" << b << endl;
          cout << "Result =" << Result <<  endl;
}
```

```
a=3
b=4
Result =7
a=3
b=4
Result =1
```

Quiz

Quiz 1, 2

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

void main(void)
{
          char i1=100, i2=300;
          char *p1,*p2;
          p2 = &i1;
          cout << *p2 << endl;
}
```

```
d
```

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


void main(void)
{
          int *p1;
          p1 = new int [5];
          *p1 = 2;
          *(++p1) = 4;
          cout << *p1 << endl;
          cout << *(p1+1) << endl;
          cout << *(p1-1) << endl;

}
```

```
4
-842150451
2
Press any key to continue
```

Quiz 3, 4

```
#include <iostream>
using namespace::std;
void main(void)
{
          int i = 100;
          int *ptr;
          *ptr = 100;
          cout << ptr << endl;
          cout << *ptr << endl;
}
```

```
Runtime error. Printing *ptr without initializing it
first
```

```
#include <iostream>
using namespace::std;
int *p;

void SthWrong()
{
          int b = 3;
          *p = b;
}


void main(void)
{
          p = new int;
          cout << p << endl;
          cout << &p << endl;
          SthWrong();
          cout << p << endl;
          cout << &p << endl;
          cout << *p << endl;
}
```

```
005B1DE0
00310978
005B1DE0
00310978
3
```

Quiz 5, 6

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          int Arr[10];
          for (int i = 0; i< 10; i++)
          {
                    Arr[i] = 2*i;
          }
          int* p = Arr;
          cout << *p << endl;
          cout << *p+1 << endl;
          cout << *p++ << endl;
          cout << *++p << endl;
          cout << *(p+1) << endl;
          cout << *(p+5) << endl;
}
```

```
0
1
0
4
6
14
```

```
#include <iostream>
using namespace::std;
const int ArrSize = 10;

void main(void)
{
          char KoKo[] = "WOWAIOE"; // defining a string
          char *Ptr = KoKo;
          *Ptr = 'C';
          *Ptr++ = 'K';
          *Ptr = 'B';
          cout << KoKo << endl;
}
```

```
KBWAIOE
Press any key to continue
```