Arrays, Pointers, and Struct

Yaping Jing

CS270 – Computer Science II

HW3 - Reading Files

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

// to create a file object
ifstream fin;

// to open a file
fin.open("phonenum.txt");

// to check if it is the end of of file
while(!fin.eof()){ //...
}

// to read text from file and store into variable
int x;
fin >> x;

fin.close(); // to close the file handler
```

Exercise

```
//pre: tmp is an integer array with size set to size
//post: all elements of tmp are initialized to
//      the value of size-1.

void setValue(int tmp[], int size);
```

Character Arrays and String

char ch[] = {'a', 'b', 'd', '\0' };

char ch[] = "abd";

Null Character '\0'

char ch[] = {'a', 'b', 'd', '\0' };

char ch[] = "abd";

Null Character '\0' Matters

Need to create one more space for the null character '\0'.

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

int main(){
   char ch[] = {'a', 'b'};
   cout << ch << endl;
   return 0;
}
```

Problem of output?

Null Character '\0'

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

int main(){
   char ch[] = {'a', 'b', '\0'};
   cout << ch << endl;
   return 0;
}
```

Comparing strings

Since strings are actually character arrays, can we do comparison on two string like:

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

int main(){
  char ch[] = {'a', 'b', '\0'};
  char ch2[] = "ab";
  cout << "ch==ch2? " << (ch==ch2) << endl;
  return 0;
}
```

Need to **overload operator==** for character arrays.

String Library

Since strings are actually character arrays, can we do comparison on two string like:

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

int main(){
   string s = "ab"
   string s1("ab");
   cout << "s==s1?␣" << (s==s1) << endl;
   return 0;
}
```

Two Dimensional Array

```
int A[3][2];

int A[0][0] = 2;
int A[0][1] = 20;

int A[1][0] = 10;
int A[1][1] = 11;

int A[2][0] = 26;
int A[2][1] = 31;
```

Alternative Two Dimensional Array Initialization

```
int anArray[][] =
{
   { 1, 2, 3, 4 },
   { 5, 6, 7, 8 }
};
```

Struct

A **structure** is for storing an aggregation of elements.

Unlike array, the elements of a structure can be of different types.

Struct Syntax

```
struct structure_name {
        ....
};
```

Struct Definition Example

```
struct Passenger = {
    string      name;
    MealType    mealPref;
    bool        isItalianCuisian;
    string      mealNo;
};
```

Struct Declaration and Initialization Example

```
struct Passenger = {
    string      name;
    MealType    mealPref;
    bool        isItalianCuisian;
    string      mealNo;
};
```

Passenger pass = { ”Mary Smith”, VEGETARIAN, **true**, ”29345” };

Access Struct Member Example

```
struct Passenger = {
    string      name;
    MealType    mealPref;
    bool        isItalianCuisian;
    string      mealNo;
};
```

Passenger pass = { ”Mary Smith”, VEGETARIAN, **true**, ”29345” };

```
pass.name = "Tom Hanks";
pass.mealPref = RUGULAR;
```

Exercise

```
struct Passenger = {
    string      name;
    MealType    mealPref;
    bool        isItalianCuisian;
    string      mealNo;
};
```

Passenger pass = { ”Mary Smith”, VEGETARIAN, **true**, ”29345” };

Can you modify values for other member variables such as mealPref, and mealNo?

Pointers – Motivation

int x = 5;
int y = 6;

	Variables	Contents
FFF0	x	5
FFF1	y	6

Pointer Declaration Syntax

dataType *pointerVarName;

Pointer Declaration Syntax

dataType *pointerVarName;

e.g.

int *x;

double* x;

Pointers Example

int x = 5;
int* y = & x;

	Variables	Contents
FFF0	x	5
FFF1	y	?

Pointers Example

int x = 5;
int* y = & x;

	Variables	Contents
FFF0	x	5
FFF1	y	FFF0

Derefrencing Pointer Variables

int x = 5;
int* y = & x;

	Variables	Contents
FFF0	x	5
FFF1	y	FFF0

Derefrencing Pointer Variables

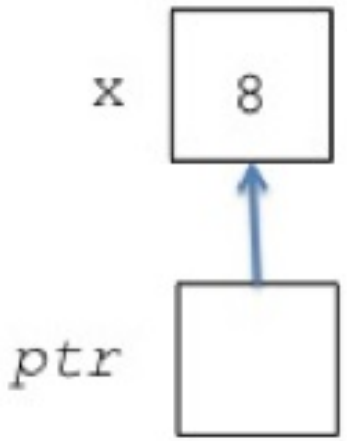
int x = 5;
int* y = & x;

	Variables	Contents
FFF0	x	5
FFF1	y	FFF0

cout << *y ;

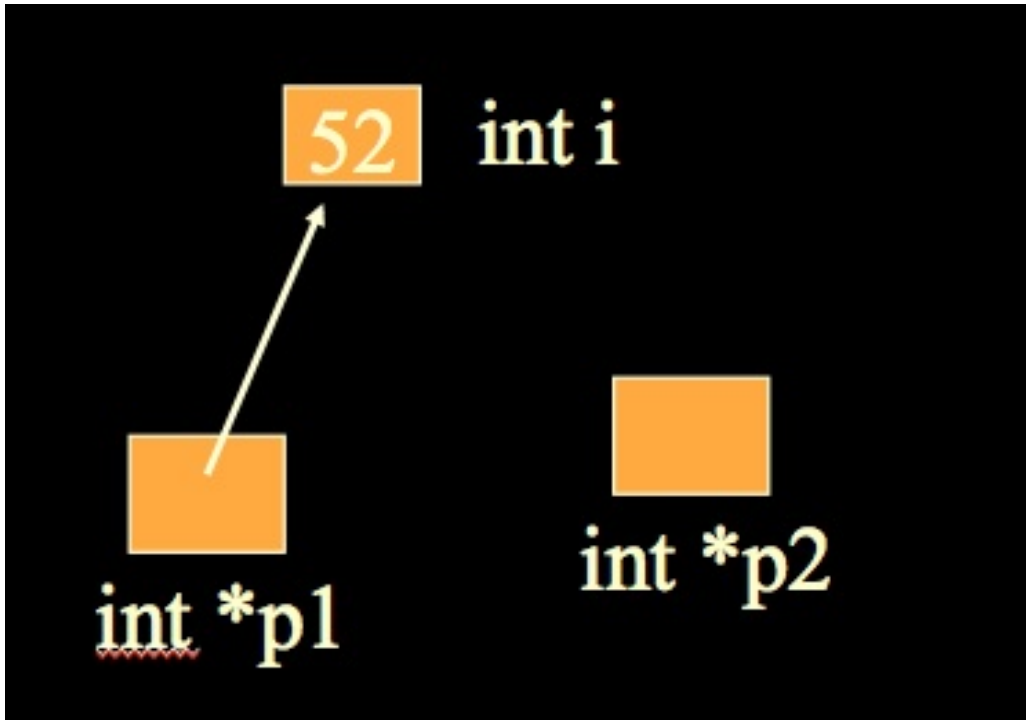
Pointer Variables

Pointer:



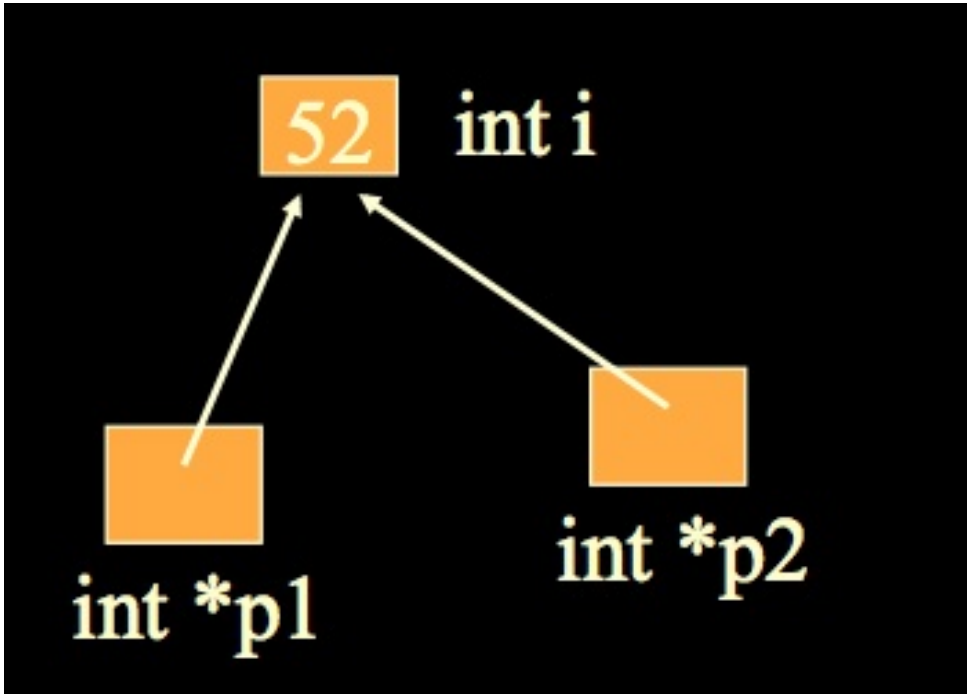

```
int x=8;
int *ptr;
ptr = &x;
```

Pointer Variables Example1



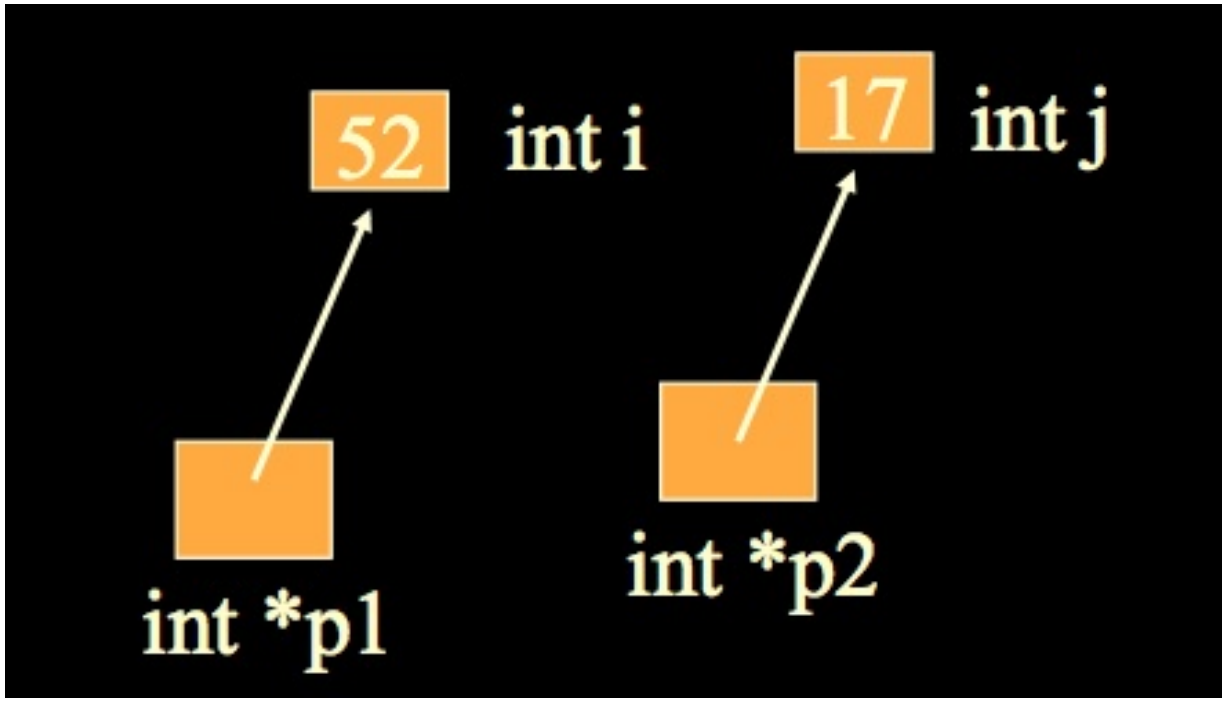

```
int i = 52;

int* p1, *p2;

p1 = &i;
```

Pointer Variables Example1



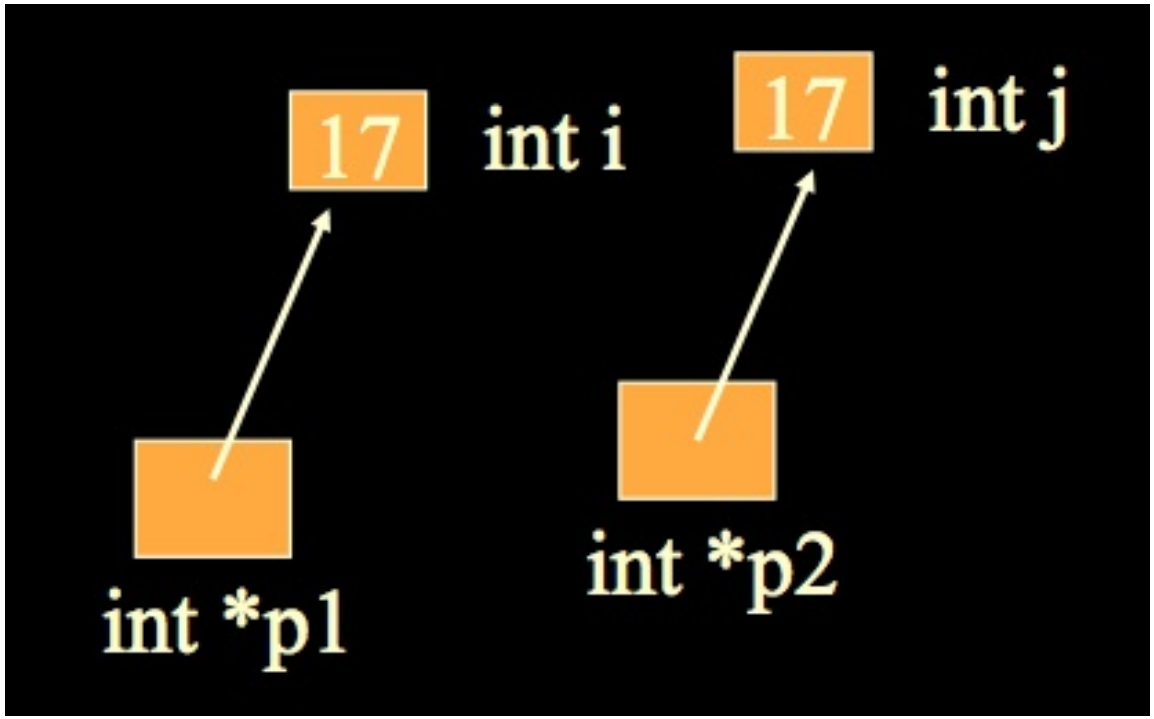

```
int i = 52;

int* p1, *p2;

p1 = &i;

p2 = p1;
```

Pointer Variables Example1




```
int i = 52;

int j = 17;

int* p1, *p2;

p1 = &i;

p2 = &j;
```

Pointer Variables Example1




```
int i = 52;

int j = 17;

int* p1, *p2;

p1 = &i;

p2 = &j;

*p1 = *p2;
```

More Pointer Variables Examples

```
char ch = 'Q';

char* p = & ch;

cout << *p;
```

More Pointer Variables Examples

```
char ch = 'Q';

char* p = & ch;

cout << *p;

ch = 'Z';

cout << *p;
```

More Pointer Variables Examples

```
char ch = 'Q';

char* p = & ch;

cout << *p;

ch = 'Z';

cout << *p;

*p = 'X';

cout << ch;
```

Pointers Initialization

```
int x = 5;
int* y = & x;
```

Pointers Initialization

int x = 5;
int* y = & x;

If a pointer is not initialized during declaration, it is wise to give it a NULL (0) value. e.g.

int *intptr = 0;

float *floatptr = NULL;

Pointers Initialization

If a pointer is not initialized during declaration, it is wise to give it a NULL (0) value. e.g.

int *intptr = 0;

float *floatptr = NULL;

Pointers Initialization

If a pointer is not initialized during declaration, it is wise to give it a NULL (0) value. e.g.

int *intptr = 0;

float *floatptr = NULL;

It is an error to dereference a pointer whose value is NULL.