UCLA Samueli
Computer Science

CS31 Week 10 Discussion

Fall 2021, Section 1C

Mingyu Derek Ma mdma@ucla.edu

Thanks Muhao Chen, Sandeep Singh Sandha, Yuanhao Xiong for their shared content

https://derek.ma/cs31 for slides and other discussion materials

Reminder

- Final exam Saturday Dec 4 6:30pm
- Solution for project 7 available

Resources for final exam review

- Final practice problems and solutions from Brian Choi, Kung-Hua Chang, Bryant Chen -> CS31 website announcement
- UPE final exam review session
- Winter 2021 practice questions

Review

If & Switch

```
if (condition1) {
        expressionA;
}
else if (condition2) {
        expressionB;
}
else{
        expressionC;
}
```

```
int vehicleClass;
double toll;
cout << "Enter vehicle class: ";
cin >> vehicleClass;
switch (vehicleClass)
{
          case 1:
                    cout << "Passenger car.";
                    toll = 0.50;
                    break;
          case 2:
                    cout << "Bus.";
                    toll = 1.50;
                    break;
          case 3:
                    cout << "Truck.";
                    toll = 2.00;
                    break;
          default:
                    cout << “Unknown vehicle
class!”;
}
```

Loops

For loop

```
int sum = 0;
for (int n = 1;n <10;n++)
        sum = sum + n;
```

For loop: need to note variable is local

while: boolean expression is checked before the loop body is executed

do-while: boolean expression is checked after the loop body is executed, always execute loop body at least once

while loop

```
while (Boolean_Expression)
{
    Statement_1
    Statement_2
      .
      .
      .
    Statement_Last
}
```

do while loop

```
do
{
    Statement_1
    Statement_2
      .
      .
      .
    Statement_Last
} while (Boolean_Expression);
```

Loops

```
for (init; condition; increment)
{
        statements;
}
```

- These three programs have the same logic

```
init
do {
        if (!condition)
                break;
        statements;
        increment;
} while (True);
```

```
init
if (condition) {
        do {
                statements;
                increment;
        } while (condition);
}
```

Function

- Declaring the function: need to specify return type,
  if no value to return use void type

```
int max(int num1, int num2);
```

- Parameters can be
  - Pass by value
  - Pass by reference

```
void getInput (double variableOne, int variableTwo);
```

```
void getInput (double &variableOne, int &variableTwo);
```

Arrays

- Declare an array

```
int c[3] = {2, 12, 1};
int b[] = {5, 12, 11};
```

- Pass the array along with the array size to a function

```
void procedure (int arr[ ], int size)
```

- Multidimensional array as an argument of a function

```
void procedure (int arr[][3][4])
```

Arrays

```
void tripler(int &n){
        n = 3 * n;
}
```

Which of the following are acceptable function calls?

```
int a[3] = {4,5,6}, number = 2;

tripler(a[2]);
tripler(a[3]);
tripler(a[number]);
tripler(a);
tripler(number);
```

Arrays

```
void tripler(int &n){
        n = 3 * n;
}
```

Which of the following are acceptable function calls?

```
int a[3] = {4,5,6}, number = 2;

tripler(a[2]);
tripler(a[3]);
tripler(a[number]);
tripler(a);
tripler(number);
```

Pointer

- Declare a pointer

```
int *p1;
double *p2;
```

- Two operations
  - *: dereference

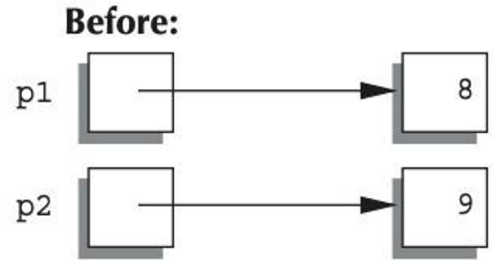
```
int *a;
// *a means retrieve the value at address a
```

  - &: reference

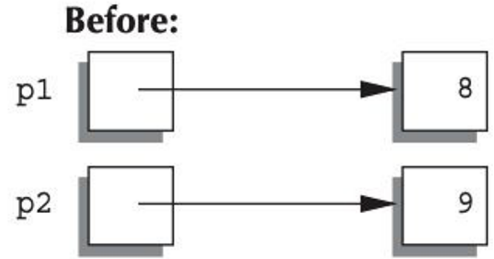
```
int v;
// &v means get the memory address of variable v
```

Pointer

```
p1 = p2;
```

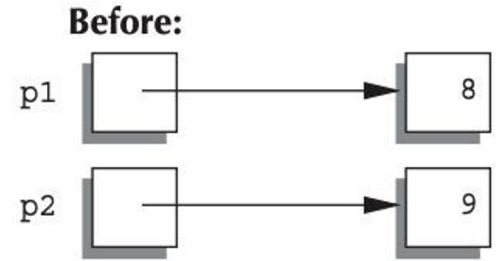


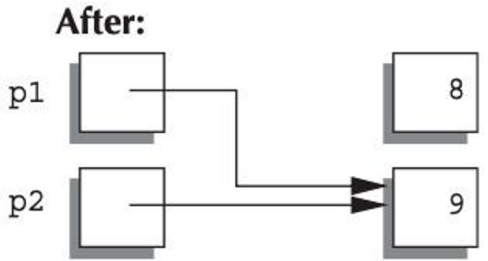

```
*p1 = *p2;
```



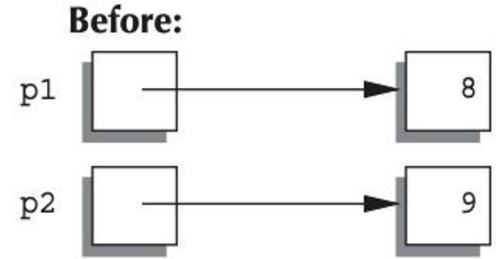
Pointer

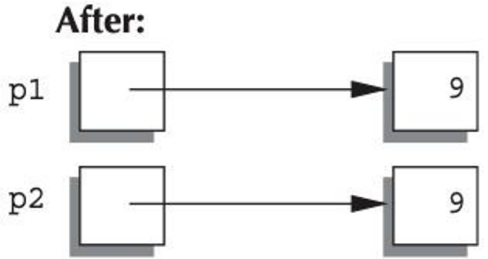
```
p1 = p2;
```






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





Struct

- struct: a collection of values of different types
- Declare a structure
  - structName instanceName;
- Access attribute
  - s1.id
  - (*p).id
  - p->id
- -> only works for pointer

```
struct student {
    string name;
    int id;
    string email;
};

student s1;
student *p = &s1;

s1.id = 389;
cout << s1.id <<endl;
cout << (*p).id <<
endl;
cout << p->id << endl
```

Pass a struct to a function

```
struct student {
  string name;
  int id;
  string email;
};

void TruncateID( student s);

int main(){
  student s1={“Yuanhao”, 123};
  TruncateID(s1);
  cout << s1.id << endl;
}
```

Output: 123

```
void TruncateID(student s) {
    if(s.id > 100)
    s.id = 100;
}
```

Pass a struct to a function

```
struct student {
  string name;
  int id;
  string email;
};

void TruncateID( student &s);

int main(){
  student s1={“Yuanhao”, 123};
  TruncateID(s1);
  cout << s1.id << endl;
}
```

Output: 100

```
void TruncateID(student &s) {
    if(s.id > 100)
    s.id = 100;
}
```

Class

- Private members: only member friends can access
- By default, members in class are private
- Constructor
  - Used to initialize member values
  - Name is the same as the class name
  - Must be a public member
  - No return type

```
class student {  public:
        // constructor
        student(int idvalue);
    private:
        string name;
        int id;
        string email;
        char grade;
};

student s1;
student *p = &s1;
```

new operator

- Creates a new dynamic variable of a specified type and returns a pointer that points to the new variable

```
MyType *p;
p = new MyType;
```

```
int *n;
n = new int(17); // initializes *n to 17
```

```
SomeClass *classPtr;
classPtr = new SomeClass; //Calls default constructor.
```

```
MyType *mtPtr;
mtPtr = new MyType(32.0, 17); // calls MyType(double, int);
```

Thank You