Lecture 2 — Chapter 1

Class Outline and C++ Review

Feb 2, 2017

`lvalue`**s and** `rvalue`**s**

I am going to lie to you. (Sorry!)

`lvalue`s and `rvalue`s

An `lvalue` can go on the left hand side of an expression, an `rvalue` cannot go on the left hand side of an expression.

`lvalue`s and `rvalue`s

An `lvalue` can go on the left hand side of an expression, an `rvalue` cannot go on the left hand side of an expression.

1. `a = b;`

`lvalue`s and `rvalue`s

An `lvalue` can go on the left hand side of an expression, an `rvalue` cannot go on the left hand side of an expression.

1. `a = b;`
2. `a = b + c;`

`lvalue`s and `rvalue`s

An `lvalue` can go on the left hand side of an expression, an `rvalue` cannot go on the left hand side of an expression.

1. `a = b;`
2. `a = b + c;`
3. `a + b = c;`

`lvalue`s and `rvalue`s

An `lvalue` can go on the left hand side of an expression, an `rvalue` cannot go on the left hand side of an expression.

1. `a = b;`
2. `a = b + c;`
3. `a + b = c;`
4. `int *a = &c;`

`lvalues` and `rvalues`

An `lvalue` can go on the left hand side of an expression, an `rvalue` cannot go on the left hand side of an expression.

1. `a = b;`
2. `a = b + c;`
3. `a + b = c;`
4. `int *a = &c;`
5. `int *a = &(b + c);`

C++ Review (Pointers & References)

C++ Review (Pointers & References)

What are the differences?

C++ Review (Pointers & References)

What are the differences?

1. You cannot initialize a reference with "`nullptr`".

C++ Review (Pointers & References)

What are the differences?

1. You cannot initialize a reference with "`nullptr`".
2. A reference can only be assigned a value during initialization.

C++ Review (Pointers & References)

What are the differences?

1. You cannot initialize a reference with "`nullptr`".
2. A reference can only be assigned a value during initialization.
3. A reference only offers a single level of indirection.

```cpp
#include <iostream>

int& foo(int &b)
{
    return b;
}

int main()
{
  int bar = 1;
  foo(bar) = 42;
  int *ptr = &foo(bar);
  std::cout << "ptr: " << *ptr << std::endl;
  std::cout << "bar: " << bar << std::endl;
  return 0;
}
```

C++ Review (Copy Constructors)

What's the problem?

C++ Review (Copy Constructors)

```
1  struct File {
2      char *name;
3      bool can_read;
4      bool can_write;
5      File(char *name, bool r=false, bool w=false)
6          : can_read(r), can_write(w)
7      {
8          this->name = strdup(name);
9      }
10 }
11 File foo("foo");
12 File bar = foo;
```

C++ Review (Copy Constructors)

What's a solution?

C++ Review (Copy Constructors)

We can add the following copy constructor to `File`.

```
1  File(const File& file)
2  {
3      name = strdup(file.name);
4      can_read = file.can_read;
5      can_write = file.can_write;
6  }
```

C++ Review

What about when the objects go out of scope?

C++ Review (Destructors)

```
1  ~File()
2  {
3      if (name != nullptr) {
4          free(name);
5      }
6  }
```

What is an Abstract Data Type?

What is an Abstract Data Type?

An abstract data type is defined by its "behavior", from the point of view of a user of the data type.

What is an Abstract Data Type?

An abstract data type is defined by its "behavior", from the point of view of a user of the data type.

What is meant by "behavior"?

What is an Abstract Data Type?

An abstract data type is defined by its "behavior", from the point of view of a user of the data type.

What is meant by "behavior"?

1. Possible values.

What is an Abstract Data Type?

An abstract data type is defined by its "behavior", from the point of view of a user of the data type.

What is meant by "behavior"?

1. Possible values.
2. Possible operations.

What is an Abstract Data Type?

An abstract data type is defined by its "behavior", from the point of view of a user of the data type.

What is meant by "behavior"?

1. Possible values.
2. Possible operations.
3. Possible errors.

Abstract Data Types

Examples

Abstract Data Types

Examples

1. Car

Abstract Data Types

Examples

1. Car
2. Phone

Abstract Data Types

Examples

1. Car
2. Phone
3. Student

Abstract Data Types

Examples

1. Car
2. Phone
3. Student