ESc 101: Fundamentals of Computing

Lecture 33

Apr 1, 2010

Outline

Global Variable

- A global variable is defined outside of any function.
- Its scope is over the entire program from the point it is defined.
- In the recursive function for Fibonacci sequence, we use two global variables: `F[]` and `m`.

Global Variable

- A global variable is defined outside of any function.
- Its scope is over the entire program from the point it is defined.
- In the recursive function for Fibonacci sequence, we use two global variables: `F[]` and `m`.

The SIZE Variable

- To fix the size of each number during execution, we need to make the `SIZE` also variable instead of a constant.
- One possible way of doing this is to defined it in one of the files for addition library.
- However, it will not then be available in the files preceding it in during compilation.

The SIZE Variable

- To fix the size of each number during execution, we need to make the `SIZE` also variable instead of a constant.
- One possible way of doing this is to defined it in one of the files for addition library.
- However, it will not then be available in the files preceding it in during compilation.

The SIZE Variable

- To fix the size of each number during execution, we need to make the `SIZE` also variable instead of a constant.
- One possible way of doing this is to defined it in one of the files for addition library.
- However, it will not then be available in the files preceding it in during compilation.

Defining SIZE in Header File

- To get around this, we can define `SIZE` in the header file `numbers.h`.
- However, since the header is included in every file, this gives rise to multiple places where `SIZE` is defined: this is not allowed!
- Is there a way to circumvent this problem?

Defining SIZE in Header File

- To get around this, we can define `SIZE` in the header file `numbers.h`.
- However, since the header is included in every file, this gives rise to multiple places where `SIZE` is defined: this is not allowed!
- Is there a way to circumvent this problem?

Defining SIZE in Header File

- To get around this, we can define `SIZE` in the header file `numbers.h`.
- However, since the header is included in every file, this gives rise to multiple places where `SIZE` is defined: this is not allowed!
- Is there a way to circumvent this problem?

The extern Declaration

In one of the files, we define the variable:

```
int SIZE = 10;
```

In all other files, we declare that the variable is defined in some other file by saying:

```
extern int SIZE;
```

The extern Declaration

In one of the files, we define the variable:

```
int SIZE = 10;
```

In all other files, we declare that the variable is defined in some other file by saying:

```
extern int SIZE;
```

Can it be Put in the Header?

- This works, but requires `SIZE` to be defined or declared in every file.
- It is more convenient to put the definition in the header file.
- But then we come back to the problem of multiple definitions.
- We need to define `SIZE` in the header file, and also make sure that this definition is valid in only one of the files.
- In the remaining files, it is just declared using `extern`.

Can it be Put in the Header?

- This works, but requires `SIZE` to be defined or declared in every file.
- It is more convenient to put the definition in the header file.
- But then we come back to the problem of multiple definitions.
- We need to define `SIZE` in the header file, and also make sure that this definition is valid in only one of the files.
- In the remaining files, it is just declared using `extern`.

Can it be Put in the Header?

- This works, but requires `SIZE` to be defined or declared in every file.
- It is more convenient to put the definition in the header file.
- But then we come back to the problem of multiple definitions.
- We need to define `SIZE` in the header file, and also make sure that this definition is valid in only one of the files.
- In the remaining files, it is just declared using `extern`.

Can it be Put in the Header?

- This works, but requires `SIZE` to be defined or declared in every file.
- It is more convenient to put the definition in the header file.
- But then we come back to the problem of multiple definitions.
- We need to define `SIZE` in the header file, and also make sure that this definition is valid in only one of the files.
- In the remaining files, it is just declared using `extern`.

The ifdef Directive

We solve this by differentiating between files using directives to the compiler:

<2->
```
#ifdef SOME_CONSTANT
    int SIZE = 10;
#endif
```

The above tells the compiler to include the variable definition in a file only if the constant `SOME_CONSTANT` is defined in the file.

The ifdef Directive

We solve this by differentiating between files using directives to the compiler:

<2->
```
#ifdef SOME_CONSTANT
    int SIZE = 10;
#endif
```

The above tells the compiler to include the variable definition in a file only if the constant `SOME_CONSTANT` is defined in the file.

The ifdef Directive

We solve this by differentiating between files using directives to the compiler:

<2->
```
#ifdef SOME_CONSTANT
    int SIZE = 10;
#endif
```

The above tells the compiler to include the variable definition in a file only if the constant `SOME_CONSTANT` is defined in the file.

The ifndef Directive

```
#ifndef SOME_CONSTANT
    extern int SIZE;
#endif
```

The above tells the compiler to include the variable declaration in a file only if the constant `SOME_CONSTANT` is not defined in the file.

The ifndef Directive

```
#ifndef SOME_CONSTANT
    extern int SIZE;
#endif
```

The above tells the compiler to include the variable declaration in a file only if the constant `SOME_CONSTANT` is not defined in the file.

Putting it Together in the Header File

```
#ifdef NUMBER_SIZE
    int SIZE = 10;
#endif
#ifndef NUMBER_SIZE
    extern int SIZE;
#endif
```

Declarations in Other Files

- In one of the files, we give
  #define NUMBER_SIZE 1 before including the header file.
- In all other files, we do nothing.

Declarations in Other Files

- In one of the files, we give
  #define NUMBER_SIZE 1 before including the header file.
- In all other files, we do nothing.