Chapter - 23 Modular Programming

Modules

A module is a collection of functions or classes that perform related functions.
If a program is a book, a module is a chapter.
Modules consist of a public and private part.

Public Part

- Defines how the module is to be used
- Is put in a header file to be used by other people

Private Part

- Does the work
- Contains the “details” that the user does not have to worry about.
- Is put in a C++ source file.

extern modifier

Is used to indicate a function or variable defined in another file.

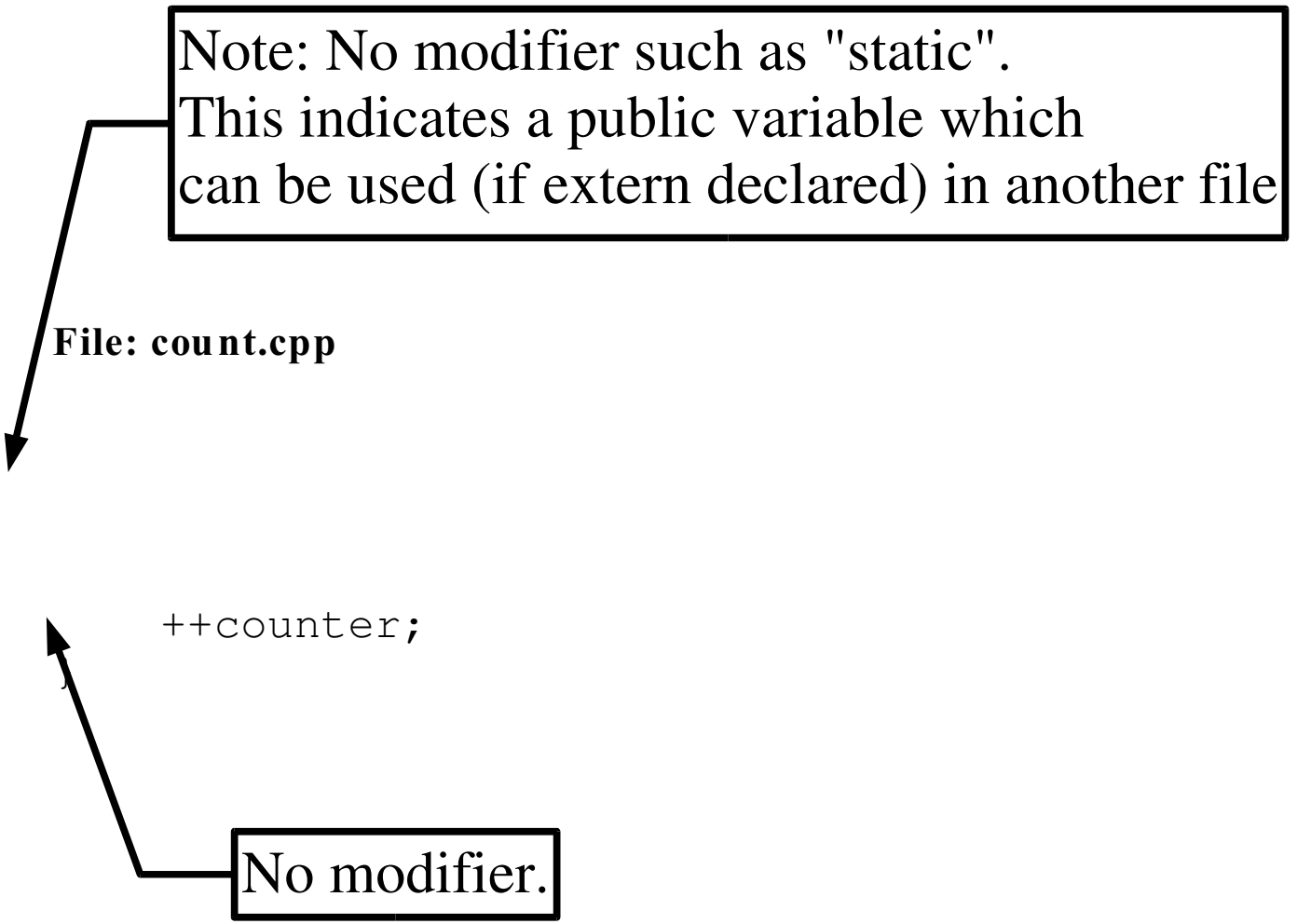
File: main.cpp

extern

extern

```
int main() {


        inc_counter();


}
```

extern modifier



Modifiers for Global Data

Modifier	Meaning
`extern`	Variable/function is defined in another file.
<blank>	Variable/function is defined in this file (public) and can be used in other files.
`static`	Variable/function is local to this file (private).

Examples

The following is legal:

The following is legal, but causes a lot of problems. (Some linkers check for this and scream when the see it.)

File: main.cpp

```
main(){

    // ....
```

File: sub.cpp

static is OK

File: main.cpp

```
int main() {

}
```

File: sub.cpp

Headers, the public part

Headers contain:

- A comment section describing clearly what the module does and what is available to the user.
- Public class definitions
- Common constants.
- Public structures.
- Prototypes of all the public functions.
- extern declarations for public variables.

File: ia.h

```
/*************************************************************************

 *-----------------------------------------------------------------------*

 *************************************************************************/
```

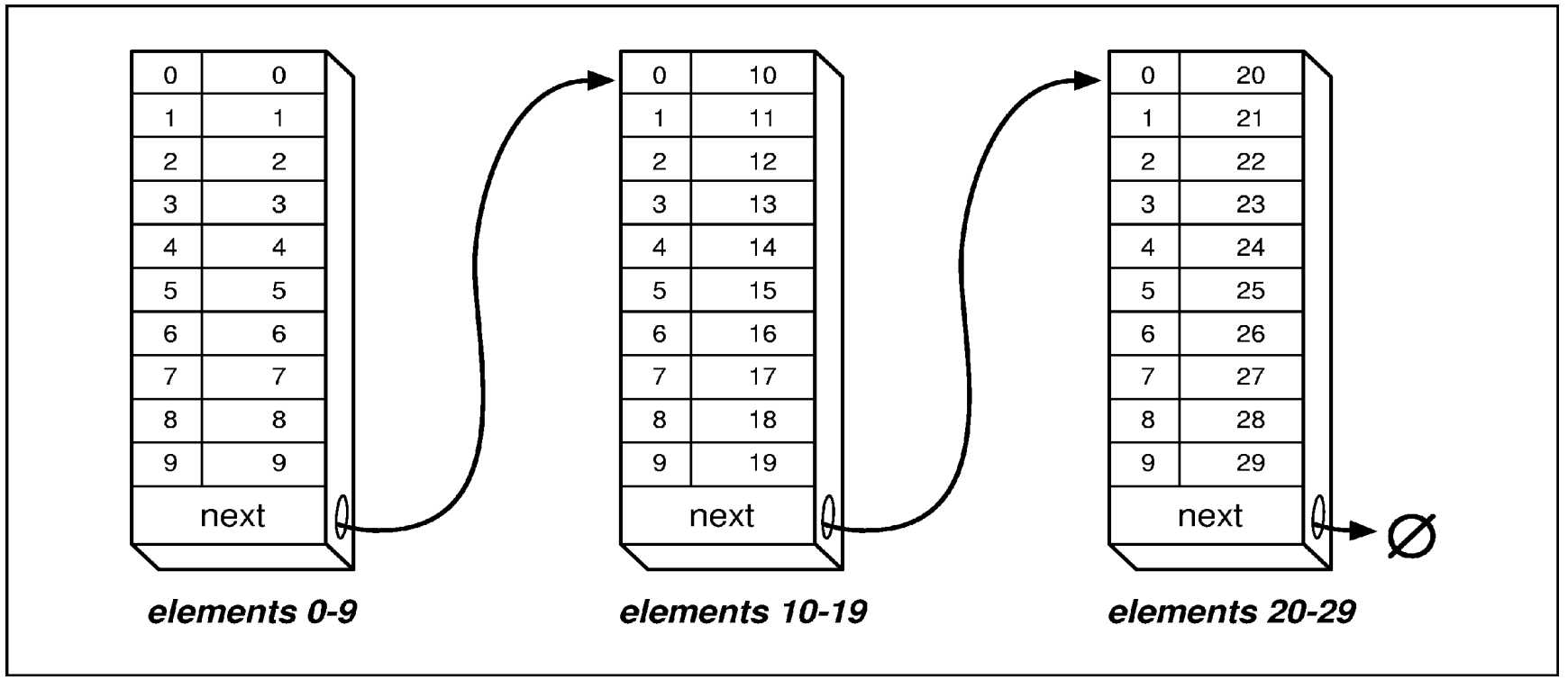
File: ia.h (continued)

```
    private:



    public:


        }

        ~infinite_array(void);


};
```

Infinite Array Code

Note: In order to use the Infinite array, all you need is the header. The real work is done in the body of the program which you don't need to see.
All functions and variables in the body that are not public are declared static to hide them from the outside world.

Infinite array structure



ia.cpp

```
/**************************************************************************

**************************************************************************/
```

ia.cpp (continued)

```
/*************************************************************

*************************************************************/
```

ia.cpp (continued)

```
                    exit(8);
                }
            }


        }

    }
```

ia.cpp (continued)

```
/*************************************************************

 *************************************************************/
infinite_array::~infinite_array(void)
{
    /*

     *




     */



    }
}
```

Makefile for GNU g++ command

```
clean:
```

Using the Infinite array

File: hist.cpp

```
/**************************************************************************

 **************************************************************************/
```

hist.cpp (continued)

```
/*

  */


/*


  */
```

hist.cpp (continue)

```
{
        exit(8);
    }

    read_data(argv[1]);
    print_histogram();

}
```

```
/*************************************************************



 *************************************************************/
{



        exit(8);
    }



           break;

        ++data_items;
    }
}
```

hist.cpp (continued)

```
/**************************************************************
 **************************************************************/
{
```

hist.cpp (continued)

```
            ++out_of_range;

            ++counters[count_index];

        }
    }
```

hist.cpp (continued)

```
                ++char_index)

        }

    }
```