Functions and the stack

Michael Nowak

Texas A&M University

Acknowledgement: Lecture slides based on those created by Bjarne Stroustrup for use with his textbook

Notes

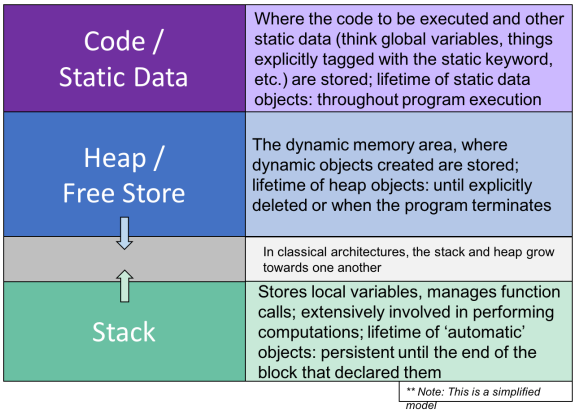
Overview

Notes

Overview

Notes

Anatomy of a program in memory



Notes

Overview

Notes

The stack

- During the execution of your programs, the stack manages function calls that are made
- Each time a function is called, an `activation record` for that function is `pushed` (added) to the stack
- The `activation record` is responsible for storing:
  - Any necessary house-keeping information (such as return location)
  - The actual arguments passed to the function
  - The local variables defined in that function
- When the function returns to its callee, its `activation record` is `popped` (removed) from the stack

Notes

The stack

- When you compile your code, the compiler examines the `variables` that will reside on the stack when a respective function is called
- The amount of space that is required for each activation record is known up front
- When a respective function is called, that amount of memory will be "allocated" on the stack
- When that same function has finished executing, the memory associated with its activation record will be "deallocated"
- This is why local variables are known as `automatic variables`: their memory is managed automatically by the function call mechanism

Notes

Overview

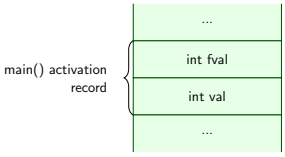
Notes

Simplified example

- When you run your program, an `activation record` for `main` is pushed for the stack

```
int main()
{
    cout << "Number:  ";
    int val;
    cin >> val;
    int fval = fact(val);

    return 0;
}
```

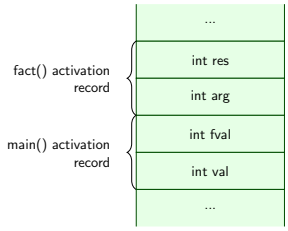


Notes

Simplified example

- When our factorial function `fact` (`int fact(int val)`) is called, an activation record for it is pushed (added) to the stack

```
int fact(int val)
{
    int res = 1;
    while(val > 1) {
        res *= val;
        val -= 1;
    }
    return res;
}
```



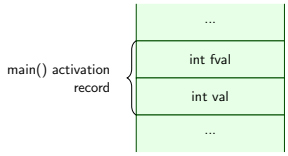
Notes

Simplified example

- When our factorial function (`int fact(int val)`) has finished executing, its activation record is popped (removed) from the stack

```
int main()
{
    cout << "Number:  ";
    int val;
    cin >> val;
    int fval = fact(val);

    return 0;
}
```



Notes

Overview

Notes

Where's my program code stored?

- The code defining each function (including main) is stored in the `code/static` region of your program's address space
- Calling a respective function retrieves the instructions from this region of memory for execution
- Meanwhile, the stack will maintain the following in its activation record:
  - Any necessary house-keeping information (such as return location)
  - Any arguments passed to the function
  - Any local variables variables that are declared in the function body

Notes

Overview

Notes

References

- Lippman, B., Lajoie, Josee, & Moo, B. E. (2016). *C++ primer* (5th ed.). Addison-Wesley.
- Stroustrup, B. (2014). *Programming: principles and practice using C++* (2nd ed.). Addison-Wesley.

Notes