Compiler Construction
PA01: Simple C Programs

Terence Henriod

August 30, 2015

Abstract

This is a demonstration of some simple C programs that are ideal candidates for testing a basic C compiler.

1 The Code

1.1 Hello World!

```
#include <stdio.h>

int
main(int argc, char** argv) {
  printf("Hello World!\n");
  return 0;
}
```

1.2 Bubble Sort

```
const int n_items = 5;

void
bubble_sort(int* items, int num_items);

int
main(int argc, char** argv) {
  int* items = (int*) malloc(n_items * sizeof(int));
  int i;
  for (i = 0; i < n_items; i++) {
    items[i] = n_items - i;
  }

  bubble_sort(items, n_items);

  free(items);
  items = 0;

  return 0;
}


void
bubble_sort(int* items, int num_items) {
  // This is the 'textbook' implementation and not the awesome optimized one
  int i = 0;
  int j = 0;
  int temp;

  for (i = 0; i < n_items; i++) {
    for (j = i + 1; j < n_items; j++) {
      if (items[i] > items[j]) {
        temp = items[i];
        items[i] = items[j];
        items[j] = temp;
      }
    }
  }
}
```

1.3 Iterative Factorial

```
int
main(int argc, char** argv) {
  int x = 5;

  int result = iterative_factorial(x);

  return 0;
}

int
iterative_factorial(int x) {
  int result = 1;

  if (x < 0) {
    result = -1;
  } else {
    while (x > 1) {
      result *= x;
      x--;
    }
  }

  return result;
}
```

1.4 Recursive Factorial

```
int
main(int argc, char** argv) {
  int x = 5;

  int result = recursive_factorial(x);

  return 0;
}

int
recursive_factorial(int x) {
  int result;

  if (x < 0) {
    result = -1;
  } else if (x <= 1) {
    result = 1;
  } else {
    result = x * recursive_factorial(x - 1);
  }

  return result;
}
```