Thomas Nguyen
COEN 171
T R 5:40 – 7:20

Homework #1

1.13 Was the first high-level programming language you learned implemented with a pure interpreter, a hybrid implementation system, or a compiler?

The first high-level programming language I learned was C. And C is implemented with a compiler. That means the source code is dissected into Lexical Units + Parse Trees into Intermediate Code. Then the Code Generator turns that into Machine Language.

1.15 How do type declaration statements for simple variables affect the readability of a language, considering that some languages do not require them

Having type declarations makes it clear exactly what type a variable is to any user that needs to understand or build upon the already existing code. This ensures that there is no confusion about what a variable's definition is.

1.18 Many contemporary languages allow two kinds of comments: one in which delimiters are used on both ends, (multiple-line comments), and one in which a delimiter marks only the beginning of the comment (one-line comments). Discuss the advantages and disadvantages of each of these with respect to our criteria

Readability: By offering multiple-line comments and one-line comments, the program becomes increasingly readable for users because the comments can be formatted to fit however much that needs to be written

Writability: Having delimiters used for multiple-line comments and one-line comments help distinctly identify which commenting style you would like to write. Additionally, having both options is helpful for whatever option is more useful to format your text

Reliability: Commenting is a very stable function offered across all languages

Cost: None

2.1 To understand the value of records in a programming language, write a small program in a C-based language that uses an array of structs that store student information, including name, age, GPA as a float, and grade level as a string (e.g., "freshmen", etc.). Also, write the same program in the same language without using structs

```
//
//  1a.cpp
//  Homework 1
//
//  Created by Thomas Nguyen on 9/27/16.
//  Copyright © 2016 Thomas Nguyen. All rights reserved.
//

#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main(int argc, const char * argv[]) {
    // Declare the structure to hold data of each individual student
    struct student {
        string name;
        int age;
        float GPA;
        string gradeLevel;
    };

    vector <student> array;

    // Declare a new Student
    student Bobby;
    Bobby.name = "Bobby Wagner";
    Bobby.age = 19;
    Bobby.GPA = 4.0;
    Bobby.gradeLevel = "Junior";

    // Place him into the array of students
    array.push_back(Bobby);

    // Print out the contents of our array of students
    vector <student>::iterator it;
    for (it = array.begin(); it<array.end(); it++) {
        cout << (*it).name << endl;
        cout << (*it).age << endl;
        cout << (*it).GPA << endl;
        cout << (*it).gradeLevel << endl;
    }
    return 0;

}
```

```
Bobby Wagner
19
4
Junior
```

```cpp
//
//  1b.cpp
//  Homework 1
//
//  Created by Thomas Nguyen on 9/27/16.
//  Copyright © 2016 Thomas Nguyen. All rights reserved.
//

#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main(int argc, const char * argv[]) {
    // Declare the vectors to hold data of each individual student
    vector <string> name;
    vector <int> age;
    vector <float> GPA;
    vector <string> gradeLevel;


    // Add a new student named Bobby
    // This requires careful insertion and removal to all vectors to
    // ensure all information about students are aligned
    name.push_back("Bobby Wagner");
    age.push_back(19);
    GPA.push_back(4.0);
    gradeLevel.push_back("Junior");


    // Print out the contents of the arrays
    vector <string>::iterator nameIt = name.begin();
    vector <int>::iterator ageIt = age.begin();
    vector <float>::iterator gpaIt = GPA.begin();
    vector <string>::iterator gradeLevelIt = gradeLevel.begin();

    int counter = 0;
    while (counter < name.size()) {

        advance(nameIt, counter);
        cout << *nameIt << endl;

        advance(ageIt, counter);
        cout << *ageIt << endl;

        advance(gpaIt, counter);
        cout << *gpaIt << endl;

        advance(gradeLevelIt, counter);
        cout << *gradeLevelIt << endl;

        counter++;

    }

    return 0;

}
```

```
Bobby Wagner
19
4
Junior
```

2.2 To understand the value of recursion in a programming language, write a program that implements quicksort, first using recursion and then without recursion

```
//
//  2a.cpp
//  Homework 1
//
//  Created by Thomas Nguyen on 9/27/16.
//  Copyright © 2016 Thomas Nguyen. All rights reserved.
//

#include <stdio.h>
#include <iostream>

using namespace std;

// Credit to (http://www.geeksforgeeks.org/iterative-quick-sort/) for help on various functions
// throughout this program


// Swaps two integers passed into function
void swap (int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}


// This function takes an array as well as the
// l --> starting index
// h --> ending index
int partition (int arr[], int l, int h) {
    // x is the pivot
    int x = arr[h];
    int i = (l - 1);

    for (int j = l; j < h; j++) {
        if (arr[j] <= x) {
            i++;
            swap (&arr[i], &arr[j]);
        }
    }

    // Places the pivot where it needs to go
    swap (&arr[i + 1], &arr[h]);
    return (i + 1);
}


/* A[] --> Array to be sorted,
 l  --> Starting index,
 h  --> Ending index */
void quickSort(int A[], int l, int h) {
    if (l < h) {
        /* Partitioning index */
        int p = partition(A, l, h);
        quickSort(A, l, p - 1);
        quickSort(A, p + 1, h);
    }
}


// A utility function to print contents of arr
void printArr( int arr[], int n ) {
    for (int i = 0; i < n; i++)
        printf( "%d ", arr[i] );
}
```

```
int main() {
    int arr [] = {1,4,7,9,3,4,5};
    int n = sizeof(arr) / sizeof(*arr);
    quickSort(arr, 0, n-1);

    printArr(arr, 7);

    return 0;
}
```

```
1 3 4 4 5 7 9 Program ended with exit code: 0
```

```
//
//  2b.cpp
//  Homework 1
//
//  Created by Thomas Nguyen on 9/27/16.
//  Copyright © 2016 Thomas Nguyen. All rights reserved.
//

#include <stdio.h>
#include <iostream>

using namespace std;


// Credit to (http://www.geeksforgeeks.org/iterative-quick-sort/) for help on various functions
// throughout this program


// A utility function to swap two elements
void swap (int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}


// This function takes an array as well as the
// l --> starting index
// h --> ending index
int partition (int arr[], int l, int h) {
    // x is the pivot
    int x = arr[h];
    int i = (l - 1);

    for (int j = l; j < h; j++) {
        if (arr[j] <= x) {
            i++;
            swap (&arr[i], &arr[j]);
        }
    }

    // Places the pivot where it needs to go
    swap (&arr[i + 1], &arr[h]);
    return (i + 1);
}
```

```c
/* A[] --> Array to be sorted,
 l  --> Starting index,
 h  --> Ending index */
void quickSortIterative (int arr[], int l, int h) {
    // Create an auxiliary stack
    int stack[h - l + 1];

    // initialize top of stack
    int top = -1;

    // push initial values of l and h to stack
    stack[++top] = l;
    stack[++top] = h;

    // Keep popping from stack while is not empty
    while (top >= 0) {
        // Pop h and l
        h = stack[ top-- ];
        l = stack[ top-- ];

        // Set pivot element at its correct position
        // in sorted array
        int p = partition(arr, l, h);

        // If there are elements on left side of pivot,
        // then push left side to stack
        if (p - 1> l) {
            stack[++top] = l;
            stack[++top] = p - 1;
        }

        // If there are elements on right side of pivot,
        // then push right side to stack
        if (p + 1 < h) {
            stack[++top] = p + 1;
            stack[++top] = h;
        }
    }
}


// A utility function to print contents of arr
void printArr( int arr[], int n ) {
    int i;
    for ( i = 0; i < n; ++i )
        printf( "%d ", arr[i] );
}


int main() {
    int arr[] = {4, 3, 5, 2, 1, 3, 2, 3};
    int n = sizeof(arr) / sizeof(*arr);
    quickSortIterative(arr, 0, n - 1);
    printArr(arr, n);
    return 0;
}
```

```
1 2 2 3 3 3 4 5 Program ended with exit code: 0
```