Computer Science 1510

Lecture 30

March 23, 2016

Lecture Outline

- typedef
- Programming examples

Final exam:
Wednesday, April 13
12 – 2 pm
EN-1001

C: typedef

- The `typedef` command in C allows for the creation of new data type names.

- Syntax:

```
typedef data_type name;
```

  where `data_type` is any existing C data type or structure, and `name` is the new data type name to be defined.

- For example,

```
typedef char* String;
```

  allows the declaration of a string of characters as a `String`, rather than an array of `char`'s.

```
String name, class[50];
```

typedef

- typedef is often used with structures, to define a data type representing a structure. For example, we might want to have several instances of a particular structure, say time of day:

```
typedef struct time{
  int hour;
  int minute;
  int second;
} Time;
```

- Here, the type `Time` is defined as a structure which represents time in hours, minutes, and seconds. Instead of typing

```
struct time now;
```

we could simply write

```
Time now;
```

where now is an instance of a variable representing time.

Example: typedef

```
#include <stdio.h>
#include <stdlib.h>

typedef struct student{
   int id;
   char first_name[20];
   char last_name[20];
   float assign[8],mid,final,grade;
} Student;

int read_student(FILE *fp, Student *st1);

int main(int argc, char *argv[])
{
   int i,j,ret;
   int cur_size,max_size;
   float total,aavg;
   Student *class;
   Student st1;
   FILE *fp=NULL;

   fp=fopen("class.dat","r");
   if (fp==NULL) return -1;  /* Unable to open file */

   class=(Student*)malloc(sizeof(Student));
   max_size=1;  /* One student allocated */
   cur_size=0;  /* No students read */
   while ((ret=read_student(fp,&st1))==0){
      cur_size++;  /* Another student read */
      if (max_size<cur_size){ /* Allocate more memory */
         max_size*=2;
         class=(Student*)realloc(class,max_size*sizeof(Student));
      }
      class[cur_size-1]=st1;
   }
   if (ret==-1) return -1;  /* Incomplete student record read */
```

```
   fclose(fp);
   for (i=0;i<cur_size;i++){
      total=0.0;
      for (j=0;j<8;j++){
         total+=class[i].assign[j];
      }
      aavg=total/8.0;
      class[i].grade=aavg/20.0*30.0+class[i].mid/50.0*30.0+class[i].final/50.0*40.0;
      printf("%s %s received %-5.1f%% in the course\n",class[i].first_name,
             class[i].last_name,class[i].grade);
   }
   return 0;
}

int read_student(FILE *fp, Student *st1) {
   char buff[1024];

   if(fgets(buff,sizeof(buff),fp)==NULL) return 1;
   sscanf(buff,"%s %s",st1->first_name,st1->last_name);

   if(fgets(buff,sizeof(buff),fp)==NULL) return -1;
   sscanf(buff,"%d",&st1->id);

   if(fgets(buff,sizeof(buff),fp)==NULL) return -1;
   sscanf(buff,"%f %f %f %f %f %f %f %f",&st1->assign[0],&st1->assign[1],
      &st1->assign[2],&st1->assign[3],&st1->assign[4],&st1->assign[5],
      &st1->assign[6],&st1->assign[7]);

   if(fgets(buff,sizeof(buff),fp)==NULL) return -1;
   sscanf(buff,"%f",&st1->mid);

   if(fgets(buff,sizeof(buff),fp)==NULL) return -1;
   sscanf(buff,"%f",&st1->final);

   return 0;
}
```

Sample input/output

- The following is an example of a data file used for the above code:

```
John Doe
12345
19 17 15 10 16 17 18 14
35
37
Jane Smith
54321
16 18 14 13 10 11 18 16
32
38
```

- Given this data file, the following output would be obtained:

```
John Doe received 74.2 % in the course
Jane Smith received 71.3 % in the course
```