プログラミング Ⅰ

Report#4

提 出 日:2006年5月25日(木)

所 属 :工学部情報工学科

学籍番号: 065702G

氏名 : 新 垣 智 規

◎問題

変数のスコープと記憶域クラスについて考察せよ。

コンパイルは先週の make コマンドを用い、Makefile とともに実行可能ファイルが作成されるまでの実行結果を示すこと。

◎ソースコード（average+）

「average+.c」

```
/*                                                              program     : average+.c
Student-ID : 065702G                                        Author      : ARAKAKI,Tomonori
UpDate     : 2006/06/04 (Sun)                           Comments    : 結合(linkage)を用いて、
平均・差を求める。     */

#include <stdio.h>

int score[10]={3,5,8,9,10,6,7,9,8,3};
int i;
float get_ave(void);
void print_data();

int main(){

  get_ave();
  printf("\n");
  print_data();
  printf("\n");

  return(0);
}
```

「get_ave.c」

```
#include <stdio.h>
  extern score[10];
  extern int i;

float get_ave(void){

  float ave = 0.0;

  for(i=0; i<10; i++) ave = ave + (float)score[i];
  ave/=10.0;
  printf("ave = %3,1f\n",ave);

  return(0);
}
```

「print_data.c」

```
#include <stdio.h>
extern score[10];
extern int i;
float dif;

void print_data(){
  float ave;

  for(i=0; i<10; i++){
  dif = score[i]-ave;
  printf("score[%02d]=%2d Difference from average = %4.1f\n",i,score[i],dif);

 }
```

```
}
```

「Makefile の emacs」

```
#
#レポート#4 の Makefile
#

average+: average+.o get_ave.o print_data.o
          cc -o average+ average+.o get_ave.o print_data.o

average+.o: average+.c
            cc -c average+.c

get_ave.o: get_ave.c
           cc -c get_ave.c

print_data: print_data.c
            cc -c print_data.c
```

◎実行結果

```
[nw0602:~] j06002% ./average+
ave = 6.800000

score[00]= 3 Difference from average = -3.8
score[01]= 5 Difference from average = -1.8
score[02]= 8 Difference from average =  1.2
score[03]= 9 Difference from average =  2.2
score[04]=10 Difference from average =  3.2
score[05]= 6 Difference from average = -0.8
score[06]= 7 Difference from average =  0.2
score[07]= 9 Difference from average =  2.2
```

```
score[08]= 8 Difference from average =  1.2
score[09]= 3 Difference from average = -3.8
```

◎ 考察

Makefile での初のレポート。

１つのプログラムを３つに分け、それを Makefile で１つのプログラムにまとめていくという現代のプログラム制作において基本的な制作方法を学ぶことが今回のレポートの主な課題と思う。

「auto」や「static」、「extern」などの代表的な記憶域クラス指定子を使い、例題のプログラムを main 関数を含む「average+.c」、平均値を出す「get_ave.c」、指定された値と平均値の差を出す「print_data.c」の３つのプログラムに分割し、Makefile にまとめて実行してみた。

まず、main 関数を含む「average+.c」は、main の前にすべての変数を宣言して他の関数にも使いやすくし、この main の中では「get_ave.c」と「print_data.c」の結果を出力する、という意味のプログラムを書いた。

要するに、「average+.c」では、変数の宣言、他の二つの実行結果を表示するということのみをさせている。

続いて「get_ave.c」では、main で宣言した『score[10]』と『int i』を、各々「extern」を使って「average+.c」から呼び出し、その変数を使用して平均値を求めるプログラムを書いた。

また、変数 ave の宣言は get_ave 関数の中で宣言し、使用した。

最後に「print_data.c」は、get_ave.c と同じように score と i を呼び出し変数 dif を宣言、差を求めるにあたって変数 ave を関数の中で呼び出した。

get_ave の結果を含んだ変数 ave を使って、dif を計算、それを表示するというプログラムが完成した。

また、記憶域の考察だが、「static」を使うと、前の変数を置き換えてしまうことが分かった。上記のプログラムで「score[10]」を static で宣言してしまうと、すべての出力結果が０になった。

これは、main では指定しているが、他の関数の前で static を宣言しているが、その中に何も入っていない為に起こったと推測できる。だが、課題ページの例

題を見る限り static は関数内のみに影響を及ぼし、他の関数の変数には影響を及ぼさないと考察できる。
「extern」だが、これを使うことによって main で定義している変数 score や i を呼び出し、そのままの変数を使えていることより、static とは違い、extern は純粋に呼び出すことだけを意味しているように思える。
また、auto に関しては、省略されているので気がつきにくいが、ほとんどの処理の際には auto が使われている。というのも計算の際に数字を初期化しないと、正しい計算ができないからだと思われる。

◎ 感想
えっと、もうこのレポートは泣き言だらけで、逆に書く泣き言がありません。。
何をしてもエラーエラーエラー、まさにエラーだらけのレポートで、変数の宣言の仕方、その使い方や extern などの利用方法など、いろいろ試行錯誤してやっていました。
その試行錯誤のおかげで知識もつきましたが、いろいろな面できつい一週間でした・・・、これで序の口なんだから想像よりも厳しいですね。
気合いを入れ直して、これからも取り組んでいきたいと思います。

あと、泣き言といえばテスト。
真偽の「０」と「１」を完璧に間違えてたことに気付きました。
これぞ後の祭り・・・。