プログラミング1
Report#05

提出日:2009/06/18(木)

所属:工学部情報工学科
学籍番号:095736E
氏名:玉城 翔

問 1.次のプログラムは外部変数に定義され、初期化された 10 個の int 型データの平均を求め、各データと平均との差を表示するプログラムである。このプログラムを以下のような関数の翻訳単位にファイルを分け、同様な動作をするプログラムを作成せよ。

型	関数名	引数・パラメータ	機能	戻り値
Float	get_ave	なし	平均値を求め表示	平均値
Void	print_data	配列の添字、平均値	1つのデータと平均値との差を求め表示	なし

ソースプログラム：average01.c

```
---------------------------------------------------------------------------------------------
/*
programu    : average.c
student-ID  : 095736E
authar      : Kakeru TAMASHIRO
updata      : 2009/06/18/(Thu)
comment     : メインの関数
*/

#include <stdio.h>

/*変数の宣言*/
int score[10]={3,5,8,9,10,6,7,9,8,3};

/*関数の宣言*/
float get_ave();
void print_data(int i, float ave);

/*main 関数*/
int main(){

  int i;/*for 文のカウント*/
  float ave;/*平均値*/

  ave = get_ave();/*平均値を求め表示*/
  print_data(i, ave);/*カウント i にあたるデータと平均値との差を求め表示*/

return(0);

}
```

ソースプログラム：get_ave01.c

```
/*
programu    : average.c
student-ID  : 095736E
authar      : Kakeru TAMASHIRO
updata      : 2009/06/18/(Thu)
comment     : 平均値の関数
*/

#include <stdio.h>

extern int score[10];

/*get_ave 関数*/
float get_ave(){

int i;/*for 文のカウント*/
float ave;/*平均値*/

for(i = 0; i < 10; i++)/*for 文の条件*/

ave = ave + (float)score[i];/*10 個のデータの総数*/

ave = ave / 10.0;/*総数の10で割った平均値*/

printf("ave = %3.1f\n",ave);/*平均値を小数点数表示*/

return(ave);

}
```

ソースプログラム：print_data01.c

```
/*
programu    : average.c
student-ID  : 095736E
authar      : Kakeru TAMASHIRO
updata      : 2009/06/18/(Thu)
comment     : 平均値とデータの差の関数
*/

#include <stdio.h>
```

```
extern score[10];

/*print_data 関数*/
void print_data(int i, float ave){

float dif;/*平均値とデータの差*/

for(i = 0; i < 10; i++){

dif = score[i] - ave;/*カウント i にあたるデータと平均値の差*/

/*カウント i ごとに差を表示*/
printf("score[%02d]=%2d  Difference from average = %4.1f\n",i ,score[i], dif);

    }

}
```

メイクファイル：makefile

```
#
#average01 の makefile
#

/*実行可能ファイルにコンパイル*/
average: average01.o get_ave01.o print_date01.o
	cc -o average average01.o get_ave01.o print_date01.o

/*オブジェクトファイルにコンパイル*/
average01.o: average01.c
	cc -c average01.c
get_ave01.o: get_ave01.c
	cc -c get_ave01.c
print_date01.o: print_date01.c
	cc -c print_date01.c
```

出力結果

```
[nw0936:~/prog1/average] e095736% make
cc -c average01.c
cc -c get_ave01.c
cc -c print_data01.c
```

```
cc -o average average01.o get_ave01.o print_data01.o
[nw0936:~/prog1/average] e095736% ./average
ave = 6.8
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
[nw0936:~/prog1/average] e095736%
```

考察

average01.c で大まかなプログラミングを行い、後で作る細かいプログラミングの関数名を加える。こうする事により、プログラミング作業の分担化が行える様になる。

get_ave 関数の中で、10個のデータの平均値を出すのだが、その為には外部変数を定義しなければならない。

それを、extern 宣言を使って外部変数の場所を示すことにより、別の場所で定義されている外部変数を使用出来る様にする。

float 型にする事により、小数点以下の数字も表示出来る様にしている。

print_data 関数の中で、データと平均値の差を出すのだが、この関数には戻り値がないため void 型で宣言されています。

makefile を作ることにより、複数個のコンパイルを make コマンドを1回使う事によりすべてコンパイルする事が出来る。
実行可能ファイルをオブジェクトファイルに依存させ、オブジェクトファイルは C ソースプログラムファイルに依存させる事により、別々のファイルでの変更があったとしても make コマンド1回で済ませられる。

問 2.変数のスコープと記憶域クラスについて考察せよ。

1.記憶域クラスについて

<4>095736E

1-1.auto(自動変数)

・関数内部で宣言され、宣言された関数の中でのみ使用可能である。(= ローカル変数)
・関数実行中のみメモリ上のスタックに確保され、関数の実行が終了すると、メモリ上から削除される。(⇒ メモリの有効利用)
・関数が呼ばれるたびに初期化を行う。
・**auto int a;** のように宣言する。ただし、「**auto**」は通常省略され、単に「**int a;**」のように宣言する。

1-2.static(静的変数)

・プログラム実行中に常に同じ場所に配置され値を保持 → 値を保持したいときに使用。(外部変数の機能)
・関数内部で宣言され、宣言された関数の中でのみ使用可能。(自動変数の機能)
・プログラム開始処理の前に一度だけ初期化を行う。
・**static int a;** のように宣言する。
・明示的に初期化を行わない場合は、初期値は 0 になる。

1-3.extern(外部変数)

・関数外で定義され、定義以降のどの関数からでも使用可能。(= グローバル変数)
・プログラム開始処理の前に一度だけ初期化を行う。
・プログラム実行中に常に同じ場所に配置され値を保持。
・明示的に初期化を行わない場合は、初期値は 0 になる。

1-4.register(レジスタ変数)

・register変数は、CPUのレジスタに記憶されます。
・使用頻度、参照頻度の高い変数をregister変数として宣言しておくことによって、実行速度の高速化を図ることが出来ます。
・register変数で宣言された変数のアドレスを参照することは出来ません。すなわちアドレス演算子が使用できません。

1-5.typedef(型定義)

・typedef変数は、新しいデータ型を作成するときに使います。
・typedef [現存するデータ型] [新しいデータ型]; のように宣言する。

サンプルプログラムより(変数スコープ)

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

```
int   fn = 0;       /* File Scope */
void  inc_n(void);
void  dec_n(void);

main(){
  printf("befor function           fn=%2d\n\n",fn);
  inc_n();
  dec_n();
  dec_n();
  inc_n();
  printf("after function           fn=%2d\n\n",fn);
}

void inc_n(void){
  static  int bn = 0;    /* Block Scope */

  printf("  befor n++       bn=%2d  fn=%2d\n",bn,fn);
  bn++;
  fn++;
  printf("  after n++       bn=%2d  fn=%2d\n\n",bn,fn);
}

void dec_n(void){
  static  int bn = 0;    /* Block Scope */

  printf("  befor n--       bn=%2d  fn=%2d\n",bn,fn);
  bn--;
  fn--;
  printf("  after n--       bn=%2d  fn=%2d\n\n",bn,fn);
}
```

出力結果

```
befor function           fn= 0

  befor n++       bn= 0  fn= 0
  after n++       bn= 1  fn= 1

  befor n--       bn= 0  fn= 1
  after n--       bn=-1  fn= 0

  befor n--       bn=-1  fn= 0
  after n--       bn=-2  fn=-1
```

```
  befor n++        bn= 1  fn=-1
  after n++        bn= 2  fn= 0

after function           fn= 0
```

考察

　変数fnと変数bnの有効範囲について、変数fnは関数外で定義されグローバル変数である。

　変数bnは関数内の最初に定義されており、その関数内でのみ使用可能である。いわゆるローカル変数である。

　変数fnはグローバル変数である為、他の関数内で定義しなくても使用可能である。つまり、有効範囲がこのサンプルプログラム全体になっている。

　変数bnはローカル関数である為、有効範囲がそれぞれの関数内部までしかない。また、静的変数であるため、値がプラスされたり、マイナスされたりした値がそのままで保持される。

参考サイト

初心者のためのポイント学習C言語
http://www9.plala.or.jp/sgwr-t/index.html

KAZZONE STYLE
http://www.ie.u-ryukyu.ac.jp/~e065701/