秋学期にむけて

春学期ではおまじないとして扱った項目について簡単に説明しよう。詳しくは秋学期に授業します。

8. C 言語のマクロ

#で始まる命令はマクロと呼びます。ビルドの際にはまずこの部分が処理されます。ここで処理された後はこの命令の跡は残りません。このマクロを処理したことを「マクロを展開した」といいます。たくさんのマクロがあるのですが、今回は以下のマクロについて説明します。

- #include
- #define

`#include`

今までおまじないで済まされてきた `#include` 命令の種明かしをします。これまで `#include <stdio.h>` を使ってきましたが、これは文字通り 'stdio.h' というヘッダーファイルをそこに挿入することを意味します。'stdio.h' には、たとえば次のような箇所があり[1]、putchar() のマクロ定義 (putchar() は関数ではない) が記してあります。

ファイル stdio.h (部分)

```
struct __FILE {
    /* NOTE: Must match (or be a prefix of) __streambuf! */
    int _flags; /* High-order word is _IO_MAGIC; rest is flags. */
    char* _gptr; /* Current get pointer */
    char* _egptr; /* End of get area. */
    char* _eback; /* Start of putback+get area. */
    char* _pbase; /* Start of put area. */
    char* _pptr; /* Current put pointer. */
    char* _epptr; /* End of put area. */
    char* _base; /* Start of reserve area. */
    char* _ebuf; /* End of reserve area. */
    struct streambuf *_chain;
};
typedef struct __FILE FILE;
(途中省略)
extern struct _fake_filebuf __std_filebuf_0, __std_filebuf_1,
__std_filebuf_2;
#define stdin ((FILE*)&__std_filebuf_0)
#define stdout ((FILE*)&__std_filebuf_1)
#define stderr ((FILE*)&__std_filebuf_2)

#define getchar() getc(stdin)
#define putchar(x) putc((x),stdout)
```

[1]処理系によって違います

また、ライブラリ関数 sqrt を使う時には ‘#include <math.h>’ として sqrt のプロトタイプ宣言を挿入します。

ファイル名を指定している部分が <> で囲んであると C 言語のシステムが与えているファイルを意味し、” ” で囲んであると個人のユーザが作成したファイルを指定することになります。

#define
このマクロには

- #define　識別子
- #define　識別子　展開された形
- #define　識別子 (引数列)　展開された形

といった表記方法があり、

#define　識別子　展開された形

の場合、その #define が現れた以降のプログラムで 「識別子」が現れた部分を「展開された形」に置き換えます。よって次の例では COUNT は展開されません (COUNT をマクロで定義するまえに COUNT を使っているので)。

悪い例

```
 1  #include <stdio.h>
 2  main()
 3  {
 4          int i,s;
 5          s = 0;
 6          for (i = 1 ; i <= COUNT ; i++) {
 7                  s = s + i;
 8          }
 9          printf("1+2+....+%d = %d\",COUNT,s);
10  }
11  #define COUNT 10
```

9. 関数

main という名の関数

printf("%d", i); というのは printf という（組込）関数を呼ぶもので、関数を自分で定義してもよい。というより、C 言語では処理をさまざまな関数に分けて、部品を組み合わせるようにプログラムを書いていくのが普通である。

これまでのプログラムはすべて

```
main ( )
{

   .
   .
}
```

という形式で書いてきたが、main も一つの関数である。実は、一つの C 言語プログラム中に main 関数が一つのみ必ず存在しなければならず、main 関数は最初に呼ばれる関数である。

巨大な main 関数を書くことは賢明ではなく、

```
void sort (int *);

main ( )
{
    int a[10000];

    arrayinput(a);
    sort(a);
    arrayoutput(a);

}

void sort (int * b)
{
    .
    .
}
```

のように処理を関数に分けてプログラムするのが普通である。詳しくは秋学期にやります。

10. ポインタ

これまでのプログラムでは、出力では
`printf("%d", i);`
などと書く一方、入力では
`scanf("%d", &i);`
などと&を付けていた。scanf という関数は、キーボードからの入力を整数値として指定された変数に入れて返すもので、scanf を呼ぶ際はその変数の場所（アドレス）を指定する。プログラムで変数のメモリ中の場所（アドレス）を指定するためには、変数の前に & を付ける。
このようにメモリに割り付けられた変数（など）のアドレスを操作対象とすることをポインタを用いた操作といい、C 言語の難所となっている。詳しくは秋学期にやります。