プログラミング１

Report#1(再)

提出日　：2013年８月１日
所　属　：工学部情報工学科
学籍番号：e135732J
氏　名　：前城　健太郎

printf()関数による標準出力

1　例題 hello.c

1.1 ソースコード全体

```
1    /*
2      Program    : hello.c
3      Student-ID : 135732J
4      Author     : MAESHIRO,Kentaro
5      UpDate     : 2013/04/21(SUN)
6      Comment    : Used Easy Function printf()
7    */
8
9    #include <stdio.h>
10
11   int main(){
12
13     printf("Hello, C World!\n");
14
15     return(0);
16   }
```

1.2 出力結果

```
Hello, C World!
```

2 例題を参考に次のように出力せよ。

2.1 出力するメッセージを変更せよ。

2.1.1 ソースコードの一部

```
9    #include <stdio.h>
10
11   int main(){
12
13     printf("Hello, My World!\n");
14
15     return(0);
16   }
```

2.1.2 出力結果

```
Hello, My World!
```

2.1.3 考察

a)　printf 関数では、"\n"を除いた("")で囲まれたメッセージを出力できる。

2.2 同じメッセージを３回、別々の行に出力せよ。

2.2.1 printf 関数内でエンターによる改行を行う

2.2.1.1 ソースコードの一部

```
9    #include <stdio.h>
10
11   int main(){
12
13     printf("Hello, My World!
14             Hello, My World!
15             Hello, My World!\n");
16
17     return(0);
18   }
```

2.2.1.2 コンパイル時のエラー

```
report2-b.c:13:10: warning: missing terminating '"' character [-Winvalid-pp-token]
 printf("Hello, My World!
        ^
report2-b.c:13:10: error: expected expression
report2-b.c:15:29: warning: missing terminating '"' character [-Winvalid-pp-token]
         Hello, My World!\n");
                            ^
```

2.2.1.3 エラーの意味

a) 13 行目：10 文字目：終わりのダブルクォーテーションが見つからない
b) 13 行目：10 文字目：予想される表現
c) 15 行目：29 文字目：終わりのダブルクォーテーションが見つからない

2.2.1.4 考察

a) printf 関数内でのエンターによる改行はコンパイル出来なかった。
b) printf 関数では 1 行で 1 つの関数として認識されると考えられる。

2.2.2 特殊文字を使い改行を表す

2.2.2.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13   printf("Hello, My World!\nHello, My World!\nHello, My World!\n");
14
15   return(0);
16  }
```

2.2.2.2 出力結果

```
Hello, My World!
Hello, My World!
Hello, My World!
```

2.2.2.3 考察

a) 1 つの printf 関数で 3 回改行することが出来た。
b) printf 関数内では、エンターによる改行ではなく"\n"を使って改行を表すことができる。
c) 前後に文章が繋がっていても"\n"を認識して改行された。

2.2.3 複数の printf 関数で改行する

2.2.3.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13   printf("Hello, My World!\n");
14   printf("Hello, My World!\n");
15   printf("Hello, My World!\n");
16
17   return(0);
18  }
```

2.2.3.2 出力結果

```
Hello, My World!
Hello, My World!
Hello, My World!
```

2.2.3.3 考察

a) printf 関数を 3 つ使っても同じような結果を得ることができた。
b) 1 つの printf 関数の最後に改行があると、次の printf 関数はその改行された行から始まる。
c) 以上より、printf 関数内の Enter キーによる改行はコンパイラされず、改行するには特殊文字である "\n"を使わなければならない。

2.3 「Hello,」と「C World!」を別々の行に出力せよ。

2.3.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello,\nC World!\n");
14
15    return(0);
16  }
```

2.3.2 出力結果

```
Hello,
C World!
```

2.3.3 考察

a) "\n"をつかうことでうまく改行することができた。

2.4 printf("…")と printf("…\n")の違いについて延べよ。
2.4.1 "\n"を消去したコードと通常のコードを比較する
2.4.1.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(”Hello, My World!”);
14
15    return(0);
16  }
```

2.4.1.2 出力結果

```
Hello, My World!%
```

2.4.1.3 考察

a) "\n"を printf 関数から除くとメッセージの後ろに "%"が現れた。
b) "\n"を printf 関数の末尾から除いた場合、"%"が現れることがわかった。

2.4.2 複数の printf 関数
2.4.2.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(”Hello, My World!1”);
14    printf(”Hello, My World!2\n”);
15    printf(”Hello, My World!3”);
16
17    return(0);
18  }
```

2.4.2.2 出力結果

```
Hello, My World!1Hello, My World!2
Hello, My World!3%
```

2.4.2.3 考察

a) 最初（13 行目）と最後（15 行目）の printf 関数には"\n"は含まれていないが、"%"が現れたのはメッセージの末尾だけであった。
b) printf("…\n")がコードの途中にある場合、通常の改行としての効果が現れる。
c) printf("…")と printf("…\n")の違いについて、複数の printf 関数の間にある場合ではメッセージの末尾で改行するかしないかを表す。
d) メッセージの最後にある printf 関数の場合ではメッセージの末尾に"%"が現れるか現れないかを表す。

2.5 同じメッセージを３回、同一行に出力せよ。

2.5.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, My World!Hello, My World!Hello, My World!/n");
14
15    return(0);
16  }
```

2.5.2 出力結果

```
Hello, My World!Hello, My World!Hello, My World!
```

2.5.3 考察

a) 狙い通りに出力することができた。
b) "%"の表示を避けるために、メッセージの末尾に"\n"を挿入した。

2.6 次のような菱形模様(「*」を用いる)を出力せよ。

```
  *
 ***
*****
 ***
  *
```

2.6.1 １つの printf 関数で表す

2.6.1.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("  *\n ***\n*****\n ***\n  *\n");
14
15    return(0);
16  }
```

2.6.1.2 出力結果

```
  *
 ***
*****
 ***
  *
```

2.6.1.3 考察

a) １つの printf 関数内で適当なスペースと改行 "\n"を使うことでうまく出力できた
b) ただし、１つの関数だけではどのような出力になるのかは分かりづらい。

2.6.2 複数の printf 関数を使って表す

2.6.2.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("   *\n");
14    printf("  ***\n");
15    printf(" *****\n");
16    printf("  ***\n");
17    printf("   *\n");
18
19    return(0);
20  }
```

2.6.2.2 出力結果

```
  *
 ***
*****
 ***
  *
```

2.6.2.3 考察

a) 複数の printf 関数の場合でも、適当なスペースと改行 "\n"を使うことで同じような出力を得ることができた。
b) ソースコード内で出力する図形がわかるので、視覚的に編集しやすい。

2.7 「*」を用いて、自分の好きな形を出力せよ。

2.7.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("      *********\n");
14    printf("     **       **\n");
15    printf("    **         **\n");
16    printf("   **           **\n");
17    printf("  **             **\n");
18    printf(" **               **\n");
19    printf("  **             **\n");
20    printf("   **           **\n");
21    printf("    **         **\n");
22    printf("     **       **\n");
23    printf("      *********\n");
24
25    return(0);
26  }
```

2.7.2 出力結果

```
      *********
     **       **
    **         **
   **           **
  **             **
 **               **
  **             **
   **           **
    **         **
     **       **
      *********
```

2.7.3 考察

a) あえて複数の printf 関数を使ってコードを書いた。
b) それにより意図した図形を簡潔に描くことができた。

2.8 テキスト PP.50【特殊な文字(エスケープシーケンス)】について考察せよ。

2.8.1 方針

考察するエスケープシーケンスとその意味はそれぞれ下記のようなものである。

1. \b 「バックスペース」：カーソルを 1 文字左へ移動
2. \f 「フォームフィード」：次ページの先頭に移動
3. \n 「改行」：次の行へ移動
4. \r 「リターン」：カレント行の先頭に移動
5. \t 「タブ」：次のタブストップに進む（最大 8 文字分）
6. \' 「アポストロフィ」：文字 '
7. \" 「ダブルクォート」：文字 "
8. \\ 「バックスラッシュ」：文字 \
9. \nnn ：文字コード nnn（8 進数）

それぞれのエスケープシーケンスを例題(hello.c)を用いて、printf 関数内のメッセージに挿入して出力結果を考察し、どのような効果があるのかを考える。

2.8.2 :"\b"「バックスペース」について

2.8.2.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C Wo\brld!\n");
14
15    return(0);
16  }
```

2.8.2.2 出力結果

```
Hello, C Wrld!
```

2.8.2.3 考察

a) メッセージ中に"\b"を挿入すると、その直前の 1 文字が消去されて出力された。
b) "\b"の効果は「直前の 1 文字を消去する」であることがわかる。

2.8.3 ”\b”が "\n"の直後にある場合

2.8.3.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C World!\n\b");
14
15    return(0);
16  }
```

2.8.3.2 出力結果

```
Hello, C World!
```

2.8.3.3 考察

a) "\b"によって"\n"が消去され"%"が末尾に現れると予想していたが、"\b"を挿入しない場合と同じ出力がされた。

2.8.4 複数の printf 関数の間に”\b”がある場合

2.8.4.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(”Hello, C World!\n”);
14    printf(”\bHello, C World!\n”);
15
16    return(0);
17  }
```

2.8.4.2 出力結果

```
Hello, C World!
Hello, C World!
```

2.8.4.3 考察

a) 13 行目の改行"\n"が"\b"によって消去されると予想していたが、"\b"を挿入しない場合と同じ出力がされた。
b) これらの結果により、"\b"は直前の 1 文字を消去する効果を持つが、メッセージの先頭や最後では機能せず、"\n"を消去する機能も持たないことが分かった。

2.8.5 "\f"「フォームフィードについて」
2.8.5.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C Wo\frld!\n");
14
15    return(0);
16  }
```

2.8.5.2 出力結果

```
Hello, C Wo
          rld!
```

2.8.5.3 考察

a) メッセージ中に"\f"を挿入すると、上記のように"\f"以降のメッセージが下にずれて出力された。
b) "\f"の効果は、「以降のメッセージを下にずらして出力する」ことがわかる。

2.8.6 “\f ”が "\n"の直後にある場合
2.8.6.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C World!\n\f");
14
15    return(0);
16  }
```

2.8.6.2 出力結果

```
Hello, C World!

```

2.8.6.3 考察

a) "\b"の場合では"\n"の前後では機能しなかったが、"\f"は前後であっても機能することが分かった。

2.8.7 複数の printf 関数の間に”\f”がある 1 行のメッセージの場合

2.8.7.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(”Hello, C Wo\frld!”);
14    printf(”Hello, C World!\n”);
15
16    return(0);
17  }
```

2.8.7.2 出力結果

```
Hello, C Wo
          rld!Hello, C World!
```

2.8.7.3 考察

a) 複数の printf 関数による 1 行のメッセージであっても、予想通りに下にずれて出力された。
b) "\f"の効果は 1 つの printf 関数の中だけでなく、複数の printf 関数の間でも効果は続くことがわかる。
c) これらの結果により、"\f"は場所に関わらず挿入位置から直後以降のメッセージを 1 行下にずらして出力する効果を持つことが分かった。

2.8.8 "\n"「改行」について
2.8.8.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello,C Wo\nrld!\n");
14
15    return(0);
16  }
```

2.8.8.2 出力結果

```
Hello, C Wo
rld!
```

2.8.8.3 考察

a) 今まで考察してきたが、改めて効果を考察する。
b) メッセージ中に"\n"を挿入すると、その地点で改行されて出力された。

2.8.9 複数の printf 関数の場合
2.8.9.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C World!\n");
14    printf("\nHello, C World!\n");
15
16    return(0);
17  }
```

2.8.9.2 出力結果

```
Hello, C World!

Hello, C World!
```

2.8.9.3 考察

a) 最後の printf 関数の先頭に"\n"を挿入すると、上記のような出力結果を得ることができた。
b) 最初の printf 関数の最後に"\n"を挿入した場合も、同じような出力結果を得ることができた。
c) これらの結果により、"\n"は場所に関わらず挿入地点から直後以降のメッセージを改行する効果があることが分かった。

2.8.10 "\r"「リターン」について

2.8.10.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C Wo\rrld!\n");
14
15    return(0);
16  }
```

2.8.10.2 出力結果

```
rld!o, C Wo
```

2.8.10.3 考察

a) ・メッセージ中に"\r"を挿入すると、上記のような出力結果が得られた。
b) ・"\r"以降のメッセージが先頭からのメッセージへ上書きされたように見える。

2.8.11 "\r"の効果の模索とその方針

ここからの考察は、それぞれ５文字づつ、全部で 20 文字のメッセージ（AAAAABBBBBCCCCCDDDDD）を用いて、"\r"の効果を詳しく調べる

2.8.11.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("AAAAABBBBB\rCCCCCDDDDD\n");
14
15    return(0);
16  }
```

2.8.11.2 出力結果

```
CCCCCDDDDD
```

2.8.11.3 考察

a) "\r"をメッセージの真ん中へ挿入した場合、"\r"から後のメッセージが出力された。

2.8.12 "\r"の効果の模索２

2.8.12.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("AAAAABBBBBCCCCC\rDDDDD\n");
14
15    return(0);
16  }
```

2.8.12.2 出力結果

```
DDDDDBBBBBCCCCC
```

2.8.12.3 考察

a) "\r"をメッセージの真ん中から右へ５文字分だけずらして挿入した場合、最初の５文字（AAAAA）が消去され、代わりに"\r"以降の５文字（DDDDD）が最初に現れて出力された。

2.8.13 "\r"の効果の模索３
2.8.13.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("AAAAA\rBBBBBCCCCCDDDDD\n");
14
15    return(0);
16  }
```

2.8.13.2 出力結果

```
BBBBBCCCCCDDDDD
```

2.8.13.3 考察

a) "\r"をメッセージの真ん中から左へ５文字分だけずらして挿入すると、"\r"以降のメッセージだけが出力された。
b) これらの結果により、"\r"は挿入位置から直後以降のメッセージを先頭のメッセージへ上書きする効果があることが分かった。

2.8.14 "\r"の効果の模索４
2.8.14.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("AAAAABBBBBCCCCCDDDDD\n\r");
14    printf("EEEEEFFFFFGGGGGHHHHH\n");
15
16    return(0);
17  }
```

2.8.14.2 出力結果

```
AAAAABBBBBCCCCCDDDDD
EEEEEFFFFFGGGGGHHHHH
```

2.8.14.3 考察

a) 複数の printf 関数の場合の効果も確認する。
b) 今までの考察と上記の結果により、"\r"は"\n"の前後では機能しないことがわかる。

2.8.15 "\r"の効果の模索５
2.8.15.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("AAAAABBBBBCCCCCDDDDD");
14    printf("\rEEEEEFFFFFGGGGGHHHHH\n");
15
16    return(0);
17  }
```

2.8.15.2 出力結果

```
EEEEEFFFFFGGGGGHHHHH
```

2.8.15.3 考察

a) "\r"の直前、13 行目末尾の"\n"を消去した場合、予想通り"\r"以降の文字が先頭へ上書きされた。
b) 以上のことにより、複数の printf 関数の場合でも"\r"による上書きは行われるが、"\r"の前後に"\n"がある場合は機能しないことが分かった。

2.8.16 "\t"「タブ」について

2.8.16.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello,C Wo\trld!\n");
14
15    return(0);
16  }
```

2.8.16.2 出力結果

```
Hello, C Wo       rld!
```

2.8.16.3 考察

a) メッセージ中に"\t"を挿入すると、長めの空白が現れた。

2.8.17 メッセージの最初、最後に"\t"がある場合

2.8.17.1 ソースコードの一部(1)

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("\tHello, C World!\n");
14    printf("12345678901234567890123456\n");
15
16    return(0);
17  }
```

2.8.17.2 ソースコードの一部(2)

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C World!\t\n");
14    printf("12345678901234567890123456\n");
15
16    return(0);
17  }
```

2.8.17.3 出力結果(1)

```
        Hello, C World!
1234567890123456789012345
```

2.8.17.4 出力結果(2)

```
Hello, C World!
1234567890123456789012345
```

2.8.17.5 考察

a) "\t"による空白部分の長さを測るために、もう 1 つの printf 関数（14 行目）を観察する printf 関数の下に追加した。
b) "\t"をメッセージの最初に挿入した場合、半角スペース 8 文字分の長さの空白部分が先頭に現れた。
c) "\t"をメッセージの最後に挿入した場合、半角スペース 1 文字分の長さの空白部分が末尾に現れた

2.8.18 メッセージの先頭から 7 文字内における"\t"について

2.8.18.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("He\tllo, C World!\n");
14    printf("12345678901234567890123456\n");
15
16    return(0);
17  }
```

2.8.18.2 出力結果

```
He      llo, C World!
12345678901234567890123456
```

2.8.18.3 考察

a) 先頭から 3 文字目に "\t"を挿入すると半角スペース 6 文字分の空白部分が現れた。
b) 先頭から 7 文字目までに "\t"を挿入すると、"\t"直後のメッセージが先頭から 9 文字目から出力されるように空白が出力されていることがわかる。

2.8.19 先頭から 8 文字目以降における"\t"について 1

2.8.19.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C\t World!\n");
14    printf("12345678901234567890123456\n");
15
16    return(0);
17  }
```

2.8.19.2 出力結果

```
Hello, C         World!
12345678901234567890123456
```

2.8.19.3 考察

a) メッセージの先頭から 8 文字目直後に"\t"を挿入すると、半角スペース 8 文字分の空白部分が再び現れた。
b) 8 文字目～15 文字目における"\t"の機能は、1 文字目～7 文字目における機能と同じである事がわかる。

2.8.20 先頭から 8 文字目以降における"\t"について 2
2.8.20.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C W\torld!\n");
14    printf("12345678901234567890123456\n");
15
16    return(0);
17  }
```

2.8.20.2 出力結果

```
Hello, C W      orld!
12345678901234567890123456
```

2.8.20.3 考察

a) メッセージの 8 文字目から 2 文字だけ右にずらして"\t"を挿入すると、予想通りに半角スペース 6 文字分の空白部分が現れた。
b) これらの結果により、printf 関数内のメッセージには半角 8 文字分ごとにある「特別な地点」があり、"\t"は挿入された地点以降のメッセージをその「特別な地点」まで空白部分によって補う効果があることが分かった。

2.8.21 " \' "「アポストロフィ」について
2.8.21.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C Wo\'rld!\n");
14
15    return(0);
16  }
```

2.8.21.2 出力結果

```
Hello, C Wo'rld!
```

2.8.21.3 考察

a) メッセージ中に" \' "を挿入すると、挿入した地点にアポストロフィ（'）が表示された。

2.8.22 " \" "「ダブルクォーテーション」について

2.8.22.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(”Hello, C Wo\”rld!\n”);
14
15    return(0);
16  }
```

2.8.22.2 出力結果

```
Hello, C Wo”rld!
```

2.8.22.3 考察

a) メッセージ中に" \" "を挿入すると、挿入した地点にダブルクォーテーション（”）が表示された。

2.8.23 "\\"「バックスラッシュ」にいついて

2.8.23.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C Wo\\rld!\n");
14
15    return(0);
16  }
```

2.8.23.2 出力結果

```
Hello, C Wo\rld!
```

2.8.23.3 考察

a) メッセージ中に"\\"を挿入すると、挿入した地点にバックスラッシュ（\）が表示された。

2.8.24 "\nnn"について

2.8.24.1 方針

a) "n"には８進数の文字が入り、"\nnn"は文字コードを表すという。
b) テキスト p407-p408 を参考にすると、８進数で 000～037、177 には特別な文字（制御文字）、040～176 には通常の文字（図形文字）が割り当てられているようだ。
c) ほとんどは何も表示されなかったが、特別な動作が現れた 010,011,012,013,014,015,033,040,について考察する。

2.8.24.2 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("\"010\"Hello, C Wo\010rld!\n");
14    printf("\"011\"Hello, C Wo\011rld!\n");
15    printf("\"012\"Hello, C Wo\012rld!\n");
16    printf("\"013\"Hello, C Wo\013rld!\n");
17    printf("\"014\"Hello, C Wo\014rld!\n");
18    printf("\"015\"Hello, C Wo\015rld!\n");
19    printf("\"033\"Hello, C Wo\033rld!\n");
20    printf("\"040\"Hello, C Wo\040rld!\n");
21    return(0);
22  }
```

2.8.24.3 出力結果

```
"010"Hello, C Wrld!
"011"Hello, C Wo     rld!
"012"Hello, C Wo
rld!
"013"Hello, C Wo
               rld!
"014"Hello, C Wo
               rld!
rld!"Hello, C Wo
"033"Hello, C Wold!
"040"Hello, C Wo rld!
```

2.8.24.4 考察

a) \010 は \b 、\011 は \t 、 \012 は \n 、\013 は \f、\014 は \f 、\015 は \r 、\033 は直後の 1 文字消去、 \040 は半角スペースと同じ機能をもつ出力結果が得られた。

2.9 エラーについて考察せよ。

2.9.1 方針

a) これまでの考察の中で、「エンターによる改行」「 "\000"」「 "\045"による "%"」が出力できないことでエラーが出力された。
b) これらのエラーの内容から、printf 関数では"\000"、"エンターによる改行"は出力できず、"%"は特別な意味を持つことが予想される。
c) 「アポストロフィ」「ダブルクォーテーション」「バックスラッシュ」についても、printf 関数内では特別な意味を持つので、" \' "、" \" "、"\\"というエスケープシーケンスが用意されてると予想される。

2.9.2 ソースコードの一部(1)

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(" ' \n");
14
15    return(0);
16  }
```

2.9.3 ソースコードの一部(2)

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(" " \n");
14
15    return(0);
16  }
```

2.9.4 ソースコードの一部(3)

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(" \ \n");
14
15    return(0);
16  }
```

2.9.5 出力結果(1)

```
 '
```

2.9.6 出力結果（エラー）(2)

```
report2-h-i1.c:13:14: error: expected ')'
  printf(" " \n");
            ^
report2-h-i1.c:13:9: note: to match this '('
  printf(" " \n");
        ^
report2-h-i1.c:13:16: warning: missing terminating '"' character
      [-Winvalid-pp-token]
  printf(" " \n");
```

2.9.7 出力結果（エラー）(3)

```
report2-h-i3.c:13:12: warning: unknown escape sequence '\x20'
  printf(" \ \n");
          ^~
```

2.9.8 考察

a) 「ダブルクォーテーション」「バックスラッシュ」については予想通りにエラーが出力されたが、「アポストロフィ」についてはそのまま出力された。
b) 次に、"%"の表示方法について考える。

2.9.9 "%"の出力方法の模索

2.9.9.1 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(" \% \n");
14
15    return(0);
16  }
```

2.9.9.2 コンパイル時のエラー

```
report2-h-i.c:13:15: warning: invalid conversion specifier '
' [-Wformat-invalid-specifier]
  printf(" \% \n");
           ~~~^
```

2.9.9.3 考察

a) " \" "や "\\"によって" " "や "\"を出力できることから、"\"を直前に置けば出力できると予想したが、エラーが出力された。
b) 今度は "\\"によって "\"が出力されることから、"%%"によって出力できると予想する。

2.9.9.4 ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf(" %% \n");
14
15    return(0);
16  }
```

2.9.9.5 出力結果

```
 %
```

2.9.9.6 考察

a) 予想通りに "%"を出力できた。