１．printf()関数による標準出力

i.例題 hello.c

●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  }
```

●出力結果

```
Hello, C World!
```

ii.例題を参考に次のように出力せよ。

a)出力するメッセージを変更せよ。

a-1:半角英語を表示する

●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  }
```

●出力結果

```
Hello, My World!
```

●考察

・ 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と考えられる。

a-2:全角日本語を表示する

●ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("こんにちは, 私の 世界!\n");
14
15    return(0);
16  }
```

●出力結果

```
こんにちは, 私の 世界!
```

●考察

・ 全角日本語についても出力できた。

b)同じメッセージを3回、別々の行に出力せよ。

b-1:printf関数内でエンターによる改行を行う

●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  }
```

●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");
                            ^
```

●エラーの意味　※推測含む

・ 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
・ 13行目：10文字目：予想される表現
・ 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

●考察
・ 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。
・ 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。

b-2:特殊文字を使い改行を表す
●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  }
```

●出力結果

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

●考察
・ 1つのprintf関数で3回改行することが出来た。
・ 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。
・ 前後に文章が繋がっていても"\n"を認識して改行された。

b-3:複数のprintf関数で改行する
●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  }
```

●出力結果

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

●考察
・ 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。
・ 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と考えられる。
・ 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。

c)「Hello,」と「C World!」を別々の行に出力せよ。
●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  }
```

●出力結果

```
Hello,
C World!
```

●考察
・ "\n"をつかうことでうまく改行することができた。

d)printf("...")とprintf("...\n")の違いについて延べよ。
d-1:"\n"を消去したコードと通常のコードを比較する
●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  }
```

●出力結果

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

●考察
・ "\n"をprintf関数から除くとメッセージの後ろに"%"が現れた。
・ "\n"をprintf関数の末尾から除いた場合、"%"が現れると考えられる。

d-2:複数のprintf関数
●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  }
```

●出力結果

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

●考察
・ 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。
・ 全体のメッセージの末尾が"\n"による改行がされてない場合に"%"が現れると考えられる。
・ 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。
・ 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。
・ 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。

e)同じメッセージを3回、同一行に出力せよ。
●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  }
```

●出力結果

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

●考察
・ 狙い通りに出力することができた。
・ "%"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。

f)次のような菱形模様(「*」を用いる)を出力せよ。

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

f-1:1つのprintf関数で表す
●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  }
```

●出力結果

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

●考察
・ 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
・ 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。

f-2:複数のprintf関数を使って表す

●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  }
```

●出力結果

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

●考察

・ 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。

・ 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と考えられる。

g)「*」を用いて、自分の好きな形を出力せよ。

●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  }
```

●出力結果

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

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

h)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。
●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。

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

h-1A:"\b"「バックスペース」について
●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  }
```

●出力結果

```
Hello, C Wrld!
```

●考察
・ 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"の効果は「直前の1文字を消去する」と予想される。

h-1B:メッセージの最初、最後にある場合
●ソースコードの一部
①

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

②

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

●出力結果
①

```
Hello, C World!
```

②

```
Hello, C World!
```

●考察
・②については"!"が消去されると予想していたが、それぞれの場合でも"\b"を挿入しない場合と同じ結果が得られた。

h-1C:"\n"の直後にある場合
●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  }
```

●出力結果

```
Hello, C World!
```

●考察
・"\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。

h-1D:複数のprintf関数の場合
●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  }
```

●出力結果

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

●考察
・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。
・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。

h-2A:"\f"「フォームフィードについて」
●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  }
```

●出力結果

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

●考察
・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。
・"\f"の効果は、「以降のメッセージを下にずらして出力する」と考えられる。

h-2B:メッセージの最初、最後にある場合
●ソースコードの一部
①

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

②

```
9   #include <stdio.h>
10
11  int main(){
12
```

```
13    printf("Hello, C World!\f\n");
14
15    return(0);
16  }
```

●出力結果

①

```


Hello, C World!
```

②

```
Hello, C World!


```

●考察

・ メッセージの先頭と最後に"\f"がある場合、先頭にあるときは1行目、最後にある場合は2行目に空白行が出力された。

・ "\b"の場合ではメッセージの先頭や最後では機能しなかった（h-1B項）が、"\f"は機能するようだ。

h-2C:"\n"の直後にある場合

●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  }
```

●出力結果

```
Hello, C World!


```

●考察

・ h-2B②項の出力結果と同じ結果が得られた。

・ "\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。

h-2D:複数のprintf関数の場合

●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("\fHello, C World!\n");
15
16    return(0);
17  }
```

●出力結果

```
Hello, C World!

Hello, C World!
```

●考察

・ 複数のprintf関数の間でも、予想通りに2行目に空白行が出力された。

h-2E:複数のprintf関数による1行のメッセージの場合

●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  }
```

●出力結果

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

●考察
・ 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。
・ "\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と考えられる。
・ 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。

h-3A:"\n"「改行」について
●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  }
```

●出力結果

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

●考察
・ 今まで考察してきたが、改めて効果を考察する。
・ 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。

h-3B:メッセージの最初、最後にある場合
●ソースコードの一部

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

②

```
9   #include <stdio.h>
10
11  int main(){
12
```

```
13    printf("Hello, C World!\n\n");
14
15    return(0);
16  }
```

●出力結果

①

```
Hello, C World!
```

②

```
Hello, C World!
```

●考察

・ 予想通りの結果が得られた。

・ "\f"と同じように、メッセージの最初と最後であっても機能することが分かった。

h-3C:複数のprintf関数の場合

●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  }
```

●出力結果

```
Hello, C World!

Hello, C World!
```

●考察

・ 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。

・ 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。

・ 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。

h-4A:"\r"「リターン」について

●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  }
```

●出力結果

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

●考察

・ 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。

・ "\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。

h-4B:メッセージの最初、最後にある場合

●ソースコードの一部

①

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

```
16  }
```

②

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

●出力結果

①

```
Hello, C World!
```

②

```
Hello, C World!
```

●考察

・ それぞれの場合、"\r"を挿入しない場合と同じ結果が得られた。

h-4C:"\n"の直後にある場合

●ソースコードの一部

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

●出力結果

```
Hello, C World!
```

●考察

"\n"の直後にある場合でも、"\r"を挿入しない場合と同じ結果が得られた

h-4D:複数のprintf関数の場合

●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("\rHello, C World!\n");
15
16    return(0);
17  }
```

●出力結果

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

●考察
・最後のprintf関数（14行目）の先頭に挿入した場合、"\r"を挿入しない場合と同じ結果が得られた。
・最初のprintf関数（13行目）の最後に挿入した場合も同様な結果が得られた。

h-4E:"\r"の効果の模索1
●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

●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  }
```

●出力結果

```
CCCCCDDDDD
```

●考察
・"\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。

h-4F:"\r"の効果の模索2
●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  }
```

●出力結果

```
DDDDDBBBBBCCCCC
```

●考察
・"\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。

h-4G:"\r"の効果の模索3
●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  }
```

●出力結果

```
BBBBBCCCCCDDDDD
```

●考察
・"\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。
・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。

h-4H:"\r"の効果の模索4
●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  }
```

●出力結果

```
AAAAABBBBBCCCCCDDDDD
EEEEEFFFFFGGGGGHHHHH
```

●考察
・ 複数のprintf関数の場合の効果も確認する。
・ h-4D項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と考えられる。

h-4I:"\r"の効果の模索5
●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  }
```

●出力結果

```
EEEEEFFFFFGGGGGHHHHH
```

●考察
・ "\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。
・ 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。

h-5A:"\t"「タブ」について
●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  }
```

●出力結果

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

●考察
・ メッセージ中に"\t"を挿入すると、長めの空白が現れた。
・ 出力結果をターミナルから文書（今現在の場合ではGoogleChromeブラウザ）へコピー・ペーストする際に空白部分の長さが変わっているように見える。
・ このことについては最後に考察する。今は便宜的に出力結果の欄には、実際のターミナルに表示された空白部分の長さを半角スペースによって表すことにする。

h-5B:メッセージの最初、最後にある場合
●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("12345678901234567890123456789\n");
15
16    return(0);
17  }
```

②

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("Hello, C World!\t\n");
14    printf("12345678901234567890123456789\n");
15
16    return(0);
17  }
```

●出力結果
①

```
        Hello, C World!
123456789012345678901234567890
```

②

```
Hello, C World!
123456789012345678901234567890
```

●考察
・"\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。
・"\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。
・"\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

h-5C:メッセージの先頭から7文字内における"\t"について
●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("123456789012345678901234567890\n");
15
16    return(0);
17  }
```

●出力結果

```
He      llo, C World!
123456789012345678901234567890
```

●考察
・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。
・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と考えられる。

h5-D:先頭から8文字目以降における"\t"について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("123456789012345678901234\n");
15
16    return(0);
17  }
```

●出力結果

```
Hello, C         World!
123456789012345678901234
```

●考察
・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。
・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だと考えられる。

h-5E:先頭から8文字目以降における"\t"について2
●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("123456789012345678901234\n");
15
16    return(0);
17  }
```

●出力結果

```
Hello, C W      orld!
123456789012345678901234
```

●考察
・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。
・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。

h-5E:ソフトウェアごとの空白部分の長さについて
●方針
ここからは、ターミナルに出力された空白部分の表示について考察する。

●ソースコードの一部

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

●出力結果

```
        Hello, C World!
```

・ "\t"を用いて、半角スペース8文字分の空白部分をターミナルへ出力した。
・ この空白部分を様々なソフトウェア（テキスト工ディット、mi、LibreOffice、Thunderbird、Googleドキュメント）へ貼り付けて、表示された長さを測定する。

●測定結果

テキスト工ディット…4文字分
mi…4文字分
LibreOffice…6文字分
Thunderbird…2文字分
Googleドキュメント…6文字分

●考察
・ 様々なソフトウェアによって、空白部分の長さは異なるものになった。
・ 空白部分はソフトウェア毎に異なる長さに設定されていると考えられる。
・ これまでの"\t"についての考察により、8文字ごとに「特別な地点」があることがわかったが、他のソフトウェアではまた違う地点にある可能性がある。

h-6A:" \' "「アポストロフィ」について
●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  }
```

●出力結果

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

●考察
・ 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。

h-6B:メッセージの最初、最後にある場合
●ソースコードの一部
①

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

②

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

●出力結果
①

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

②

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

●考察
・ メッセージの先頭や最後であっても同じように機能した。
h-6C:複数のprintf関数の場合
●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("Hello, C World!\'\n");
15
16    return(0);
17  }
```

●出力結果

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

●考察
・ 複数のprintf関数がある場合でも問題なく出力できた。
・ これらの結果により、" \' "は挿入した場所に関係なくアポスト口フィを表示する効果を持つことが分かった。

h-7A:" \" "「ダブルクォーテーション」について
●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  }
```

●出力結果

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

●考察
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。

h-7B:メッセージの最初、最後にある場合

●ソースコードの一部

①

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

②

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

●出力結果

①

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

②

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

●考察

・ メッセージの最初と最後にある場合でも、同じように出力された。

h-7C:複数のprintf関数の場合

●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("Hello, C World!\"\n");
15
```

```
16    return(0);
17  }
```

●出力結果

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

●考察
・ 複数のprintf関数がある場合でも問題なく出力できた。
・ これらの結果により、" \" "は挿入した場所に関係なくダブルクォーテーションを表示する効果を持つことが分かった。

h-8A:"\\"「バックスラッシュ」にいついて
●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  }
```

●出力結果

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

●考察
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。

h-8B:メッセージの最初、最後にある場合
●ソースコードの一部
①

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

②

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

●出力結果

①

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

②

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

●考察

・ メッセージの最初と最後にある場合でも、同じように出力された。

h-8C:複数のprintf関数の場合

●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("Hello, C World!\\\n");
15
16    return(0);
17  }
```

●出力結果

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

●考察

・ 複数のprintf関数がある場合でも問題なく出力できた。

・ これらの結果により、"\\"は挿入した場所に関係なくバックスラッシュを表示する効果を持つことが分かった。

h-9:"\nnn"について
●方針
・"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。
・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。
・最初に制御文字である"\000"〜"\037"、"\177"の効果を検証し、次に図形文字である"\040"〜"\176"の効果を検証する。

h-9A:"\000"〜"\012"について
●ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("\"000\"Hello, C Wo\000rld!\n");
14    printf("\"001\"Hello, C Wo\001rld!\n");
15    printf("\"002\"Hello, C Wo\002rld!\n");
16    printf("\"003\"Hello, C Wo\003rld!\n");
17    printf("\"004\"Hello, C Wo\004rld!\n");
18    printf("\"005\"Hello, C Wo\005rld!\n");
19    printf("\"006\"Hello, C Wo\006rld!\n");
20    printf("\"007\"Hello, C Wo\007rld!\n");
21    printf("\"010\"Hello, C Wo\010rld!\n");
22    printf("\"011\"Hello, C Wo\011rld!\n");
23    printf("\"012\"Hello, C Wo\012rld!\n");
24
25    return(0);
26  }
```

●コンパイル時のエラー

```
report2-h-8A.c:13:29: warning: format string contains '\0' within the
string
      body [-Wformat]
  printf("\"000\"Hello, C Wo\000rld!\n");
         ~~~~~~~~~~~~~~~~~~~^~~~~~~~~~
```

●エラーの意味
・13行目:29文字目:フォーマット文字列に'\0'を含んでいます

●考察
・ソースコードの13行目に"\0"が含まれているために工ラーが出力された。
・"\000"をソースコードから削除して再び実行する。

h-9B:"\000"〜"\012"について

●ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("\"000\"Hello, C World!\n");
14    printf("\"001\"Hello, C Wo\001rld!\n");
15    printf("\"002\"Hello, C Wo\002rld!\n");
16    printf("\"003\"Hello, C Wo\003rld!\n");
17    printf("\"004\"Hello, C Wo\004rld!\n");
18    printf("\"005\"Hello, C Wo\005rld!\n");
19    printf("\"006\"Hello, C Wo\006rld!\n");
20    printf("\"007\"Hello, C Wo\007rld!\n");
21    printf("\"010\"Hello, C Wo\010rld!\n");
22    printf("\"011\"Hello, C Wo\011rld!\n");
23    printf("\"012\"Hello, C Wo\012rld!\n");
24
25    return(0);
26  }
```

●出力結果

```
"000"Hello, C World!
"001"Hello, C World!
"002"Hello, C World!
"003"Hello, C World!
"004"Hello, C World!
"005"Hello, C World!
"006"Hello, C World!
"007"Hello, C World!
"010"Hello, C Wrld!
"011"Hello, C Wo    rld!
"012"Hello, C Wo
```

```
rld!
```

●考察
・"\001"〜"\007"については効果が見られない。
・"\010"、"\011"、"\012"については、それぞれ"\b"、"\t"、"\n"と同等な効果が現れた。

h-9C:"\013"〜"\025"について
●ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("\"013\"Hello, C Wo\013rld!\n");
14    printf("\"014\"Hello, C Wo\014rld!\n");
15    printf("\"015\"Hello, C Wo\015rld!\n");
16    printf("\"016\"Hello, C Wo\016rld!\n");
17    printf("\"017\"Hello, C Wo\017rld!\n");
18    printf("\"020\"Hello, C Wo\020rld!\n");
19    printf("\"021\"Hello, C Wo\021rld!\n");
20    printf("\"022\"Hello, C Wo\022rld!\n");
21    printf("\"023\"Hello, C Wo\023rld!\n");
22    printf("\"024\"Hello, C Wo\024rld!\n");
23    printf("\"025\"Hello, C Wo\025rld!\n");
24
25    return(0);
26  }
```

●出力結果

```
"013"Hello, C Wo
                rld!
"014"Hello, C Wo
                rld!
rld!"Hello, C Wo
```

```
"016"Hello, C World!
"017"Hello, C World!
"020"Hello, C World!
"021"Hello, C World!
"022"Hello, C World!
"023"Hello, C World!
"024"Hello, C World!
"025"Hello, C World!
```

●考察
・ "\016"〜"\025"については効果が見られない。
・ "\013"、"\014"は"\f"、"\015"は"\r"と同等の効果が現れた。

h-9D:"\026"〜"\037"、"177"について
●ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
13    printf("\"026\"Hello, C Wo\026rld!\n");
14    printf("\"027\"Hello, C Wo\027rld!\n");
15    printf("\"030\"Hello, C Wo\030rld!\n");
16    printf("\"031\"Hello, C Wo\031rld!\n");
17    printf("\"032\"Hello, C Wo\032rld!\n");
18    printf("\"033\"Hello, C Wo\033rld!\n");
19    printf("\"034\"Hello, C Wo\034rld!\n");
20    printf("\"035\"Hello, C Wo\035rld!\n");
21    printf("\"036\"Hello, C Wo\036rld!\n");
22    printf("\"037\"Hello, C Wo\037rld!\n");
23    printf("\"177\"Hello, C Wo\177rld!\n");
24
25    return(0);
26  }
```

●出力結果

```
"026"Hello, C World!
"027"Hello, C World!
"030"Hello, C World!
"031"Hello, C World!
"032"Hello, C World!
"033"Hello, C Wold!
"034"Hello, C World!
"035"Hello, C World!
"036"Hello, C World!
"037"Hello, C World!
"177"Hello, C World!
```

●考察
・ "\026"〜"\032"、"\034"〜"\037"、"\177"については効果が見られない。
・ "\033"については、挿入地点の直後の1文字が消去されて出力された。

h-9E:"\040"〜"\176"について
●考察
・ ソースコードを書くと膨大になるのでここでは省略する
・ "\040"については半角スペースが出力され、"\041"以降については、それぞれに対応する図形文字が出力された。ただし、"\045"についてはコンパイル時にエラーが出力された。

h-9F:"\045"について
●ソースコードの一部

```
9   #include <stdio.h>
10
11  int main(){
12
```

```
13    printf("\"045\"Hello, C Wo\045rld!\n");
14
15    return(0);
16  }
```

●コンパイル時のエラー

```
report2-h-8f.c:13:33: warning: invalid conversion specifier 'r'
      [-Wformat-invalid-specifier]
  printf("\"045\"Hello, C Wo\045rld!\n");
                              ~~~~^
```

●エラーの意味
・ 18行目:33文字目:'r'は無効な変換指定子です

●考察
・ "\045"自体ではなく、直後の文字がエラーとして出力されている。
・ テキストp407より、"\045"は"%"を表すことがわかる。
・ "\045"は"%"を表す効果があるが、printf関数内では"%"は特別な意味を表すために、このようなエラーが出力されたと考えられる。

i)エラーについて考察せよ。
●方針
・ これまでは、「b-1項:エンターによる改行」「h-8A項:"\000"」「h-8F項:"\045"による"%"」が出力できないことでエラーが出力された。
・ 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。
・ 「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。
i-1:「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」の出力について
●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  }
```

●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  }
```

●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  }
```

●出力結果①

```
 '
```

●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");
```

●出力結果（エラー）③

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

●考察

・「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。

・次に、"%"の表示方法について考える。

h-2:"%"の出力方法の模索

●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  }
```

●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");
           ~~~^
```

●考察
・" \" "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、h-8F項による工ラーと同じエラーが出力された。
・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。

●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  }
```

●出力結果

```
 %
```

●考察
予想通りに"%"を出力できた。

-あとがき-（感想・反省）
レポートを初めて書くので具体的な書き方等をほとんど知らず、先輩方を参考にして書きました。はたしてこの量で大丈夫なのでしょうか。
レポート作成を通してターミナル、emacsの扱いにも慣れて来ました。

次にレポートを書く際にはもっと効率よく考察し、量を減らして紙の節約をしたいです。