プログラミングⅠ

REPORT#1

提出日	:	2012年5月10日
所属	:	琉球大学工学部情報工学科
学籍番号	:	125722G
氏名	:	知念 辰明

0.目次

1.printf()関数による標準出力

1.printf()関数による標準出力

i.例題 hello.c

高水準コード全体▼

```
/*
    Program  : hello.c
    Student-ID:125722G
    Author    :CHINENN,Tatsunori
    UpDate    :2012/04/21(SAT)
    Comment   :It is very easy Program with Easy
               function printf().
 */
#include <stdio.h>

int main(){

    printf( "Hello,C World!\n" );
    return(0);

}
```

実行結果▼

```
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:単純な書き換え。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Program is running correctly.\n" );
    return(0);

}
```

実行結果▼

```
Program is running correctly.
```

考察▼

・printf 関数内の、
"\n"を除いたダブルクォーテーション(")で囲まれたメッセージを
書き換えることで、プログラムが表示するメッセージを変更出来る。

a-2:いろいろな文字種を試してみる。

高水準コードの一部▼

```
#include <stdio.h>
int main(){

    printf( "a z 1 0 & # ~ = ( ) / * + - % @ $ ? ! ^ \  ;  : " \n" );
    return(0);

}
```

コンパイルの際のエラー▼

```
YP1_1.c:12:11: warning: unknown escape sequence: '\040'
YP1_1.c: In function ‘main’:
YP1_1.c:12: error: stray ‘\’ in program
YP1_1.c:12: error: expected ‘)’ before ‘n’
YP1_1.c:12:64: warning: missing terminating " character
YP1_1.c:12: error: missing terminating " character
YP1_1.c:14: warning: unknown conversion type character ‘@’ in format
YP1_1.c:14: warning: unknown conversion type character ‘@’ in format
YP1_1.c:14: error: expected ‘;’ before ‘}’
```

エラーの意味▼

・12 行目：不要なバックスラッシュがコード内にある。
・12 行目："n"の前に")"がある。
・12 行目：64 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。
・14 行目："@"を変換できません。
・14 行目："}"の前に";"がある。

考察▼

・正常にコンパイルする為にエラーが出ている箇所を訂正する。
・ダブルクォーテーションで囲まれたメッセージから、"\" ")" "@" ";" "ダブルクォーテーション(")"を消去する。
・14 行目には対応する文字がない為、12 行目から消去する。

訂正した高水準コードの一部▼

```
#include <stdio.h>
int main(){

    printf( "a z 1 0 & # ~ = ( / * + - % $ ? ! ^ : \n" );
    return(0);

}
```

コンパイルの際のエラー▼

```
YP1_1.c: In function ‘main’:
YP1_1.c:12: warning: unknown conversion type character ‘$’ in format
YP1_1.c:12: warning: unknown conversion type character ‘$’ in format
```

エラーの意味▼

・12 行目："$"を変換できません。

考察▼

・正常にコンパイルする為にエラーが出ている箇所を訂正する。

・"$"をメッセージ中から消去する。

訂正した高水準コードの一部▼

```
#include <stdio.h>
int main(){

    printf( "a z 1 0 & # ~ = ( / * + - % ? ! ^ : \n" );
    /* "$" was erased from message. */
    return(0);

}
```

コンパイルの際のエラー▼

```
YP1_1.c: In function ‘main’:
YP1_1.c:12: warning: unknown conversion type character ‘?’ in format
YP1_1.c:12: warning: unknown conversion type character ‘?’ in format
```

エラーの意味▼

・12 行目："?"を変換できません。

考察▼

・正常にコンパイルする為にエラーが出ている箇所を訂正する。

・"?"をメッセージ中から消去し、再びコンパイルを試みると
同様のエラーが"!"や"^"、":"に対しても表示されたため、
これらの文字も同様に消去する。

訂正した高水準コードの一部▼

```
#include <stdio.h>
int main(){

    printf( "a z 1 0 & # ~ = ( / * + - % \n" );
    /* some characters were erased */

    return(0);

}
```

コンパイルの際のエラー▼

```
YP1_1.c: In function ‘main’:
YP1_1.c:12: warning: unknown conversion type character 0xa in format
YP1_1.c:12: warning: unknown conversion type character 0xa in format
```

エラーの意味▼

・12 行目："0xa"を変換できません。

考察▼

・"0xa"という文字はメッセージ中に含まれていない。

・よって、ここではメッセージ中に含まれた文字を１つずつ抜き出し、
その１文字だけを出力する試行をそれぞれに対して行うことで
エラーの原因となっている文字を特定する。

試行するプログラムの高水準コード▼

```
#include <stdio.h>
int main(){

    printf( "n \n" );    /* I will change "n" into other character. */
    return(0);

}
```

結果▼

・試行を繰り返した結果、"%"が文中に含まれていると
件のエラーが表示されることが分かった。

・したがって、もとのコードから"%"を消去する。

訂正した高水準コードの一部▼

```
#include <stdio.h>
int main(){

    printf( "a z 1 0 & # ~ = ( / * + - \n" );
    /* "%" was erased */

    return(0);

}
```

実行結果▼

```
a z 1 0 & # ~ = ( / * + -
```

考察▼

・無事、コンパイルを完了し実行することができた。
・今回プログラムから消去した"%"、"!"や"?"、"^"、"\"、")"、"@"、";"、":"、"ダブルクォーテーション(")"と言った文字列は通常の方法では出力できないと思われる。
・"\"や"ダブルクォーテーション(")"については、プログラムの表記に必要な文字であることから「コードの表記が正しくない」と認識されていると予測する。

a-3:全角文字を試してみる。

高水準コードの一部▼

```
#include <stdio.h>
int main(){

    printf( "あ ア 阿 安 ① \n" );
    /* This program is to output Japanese character. */
    return(0);

}
```

実行結果▼

```
あ ア 阿 安 ①
```

考察▼

・全角文字については、機種依存文字も含め出力することができるようだ。

b.同じメッセージを3回、別々の行に出力せよ。

b-1:シンプルな改行。

高水準コードの一部▼　※print 関数の()内については実際には一行である。

```
#include <stdio.h>

int main(){

    printf( "Program is running correctly.\nProgram is running
             correctly.\nProgram is running correctly.\n" );
    return(0);

}
```

実行結果▼

```
Program is running correctly.
Program is running correctly.
Program is running correctly.
```

考察▼

・1 つの printf 関数で三行にまたがって出力することができた。

・\n を入力することで、その地点で改行をすることができる。

・直前直後に文が繋がっていても、\n は改行として認識されるようだ。

b-2:改行方法を変えてみる。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Program is running correctly.\n" );
    printf( "Program is running correctly.\n" );
    printf( "Program is running correctly.\n" );
    return(0);

}
```

実行結果▼

```
Program is running correctly.
Program is running correctly.
Program is running correctly.
```

考察▼

・printf 関数を 3 つ用いることでも、同じ結果を得ることができた。

・直前の printf 関数で改行が出力されると、
次の printf 関数の出力は次の行から始まるようだ。

b-3:題意から外れるが、改行をしないようにしてみる。（\n を消してみる。）

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Program is running correctly." );
    printf( "Program is running correctly." );
    printf( "Program is running correctly." );
    return(0);

}
```

実行結果▼　※実行結果はウィンドウサイズの為に若干の改行がされた。

```
Program is running correctly.Program is running correctly.Program is
             running correctly.%
```

考察▼

・狙い通り改行をしないことができた。

・ただし、出力結果の最後に"%"が現れてしまった。

・b-2 のコードから"\n"を消去しただけなので、原因は"\n"の有無に関わっていると思われる。

b-4:"\n"に頼らない改行の模索。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Program is running correctly.
             Program is running correctly.
             Program is running correctly.\n" );
    return(0);

}
```

コンパイルの際に出たエラー▼

```
YP2_2.c:13:11: warning: missing terminating " character
YP2_2.c: In function ‘main’:
YP2_2.c:13: error: missing terminating " character
YP2_2.c:14: error: ‘Program’ undeclared (first use in this function)
YP2_2.c:14: error: (Each undeclared identifier is reported only once
YP2_2.c:14: error: for each function it appears in.)
YP2_2.c:14: error: expected ‘)’ before ‘is’
YP2_2.c:15: error: stray ‘\’ in program
YP2_2.c:15:43: warning: missing terminating " character
YP2_2.c:15: error: missing terminating " character
YP2_2.c:18: warning: format not a string literal and no format arguments
YP2_2.c:18: warning: format not a string literal and no format arguments
YP2_2.c:18: error: expected ‘;’ before ‘}’ token
```

エラーの意味▼　※推測を含みます。

・13 行目 11 文字目での警告：ダブルクォーテーションが見つからない。

・13 行目、ダブルクォーテーションが見つからない。

・14 行目、"Program"は宣言されていない関数です。
（同じ関数についての報告は 1 回しか行われません）

・14 行目、 "is"の前に")"が含まれています。

・15 行目、"\"が見つからない。

・15 行目 43 文字目での警告：ダブルクォーテーションが見つからない。

・15 行目、ダブルクォーテーションが見つからない。

・18 行目、引数形式が文字列に対応していません。

・18 行目、"}"の前に";"が含まれています。

考察▼

・メッセージが関数または引数として認識されてしまっている。
・ダブルクォーテーションでメッセージを指定しているつもりであっても、意図しない方向へ認識されている。
・printf 関数の形が認識されていない。
・これらのことより、１つの printf 関数において通常の改行（ENTER による改行）はコンパイルすらされず、エラーを生み出すだけだと考えられる。

c.「Hello,」と「C World!」を別々の行に出力せよ。

c:b で得られた結果を元に、"\n"をメッセージ内に挿入する。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,\nC World!\n" );
    return(0);

}
```

実行結果▼

```
Hello,
C World!
```

考察▼

・改行したい場所で改行することができた。
・"\n"があると、出力結果の文末に"%"が出力されなかった。

d.printf("…")と printf("…\n")の違いについて延べよ。

d-1:例題 hello.c から"\n"を消去したコードを用意し、結果を比較する。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!" );
    return(0);

}
```

実行結果▼

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

考察▼

・例題 hello.c から"\n"を消去すると、出力結果に"%"が現れた。
・項目 b、c より"\n"を挿入するとその地点で改行が行われることが分かっている。
・"\n"を入力するしないに関わらず、一文を出力するだけでは改行としての効果は見られない。
・文中の適当な場所で"\n"を挿入し、その際の出力結果から"\n"の効果を確かめる。
・"%"の有無も確かめる。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello, C World\n!" );
    return(0);

}
```

実行結果▼

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

考察▼

・確かに、"\n"を挿入した場所で改行がされた。
・ただし、文末の"%"は変わらず出力されたままである。
・次に、複数の printf 関数の間での"\n"の効果を調べる。

d-2:2 つのプログラムによる比較。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!" );
    printf( "Hello,C World!" );
    return(0);

}
```

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\n" );
    printf( "Hello,C World!" );
    return(0);

}
```

実行結果▼

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

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

考察▼

・複数の printf 関数の間であっても改行は出力される。

・左のコードでは、一個目の printf 関数には"\n"が含まれていないのに
実行結果では 2 文の間に"%"が出力されていない。

・どちらも、実行結果ではメッセージの末尾に"%"が表示されている。

・printf("......")と printf("......\n")の違いについて、
一行だけを出力するときでは"%"が表示されるかされないかとして表れる。

・複数行を出力するときには、改行として効果が表れる。

e.同じメッセージを 3 回、同一行に出力せよ。

e-1:1 つの printf 関数で出力してみる。（b-3 で行った試行とほぼ同一）

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World! Hello,C World! Hello,C World!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・"%"の表示を避けるために、2 つ目のダブルクォーテーションの前に"\n"を挿入している。

・狙い通り、同一行に同じメッセージを 3 回出力することができた。

e-2:3 つの printf 関数で同一行に出力してみる。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World! " );
    printf( "Hello,C World! " );
    printf( "Hello,C World!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・"%"の表示を避けるために、2 つ目のダブルクォーテーションの前に"\n"を挿入している。

・1 つ目、2 つ目の printf 関数では"\n"を入力しないことで、同一行にメッセージを表示することができた。

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 関数で出力してみる。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "      *\n     ***\n    *****\n     ***\n      *\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・1 つの printf 関数では、適度なスペースをそれぞれの"*"の
直前に挿入することで出力したい模様を出力することができた。

・改行（\n）は、改行をするだけで文字列をずらすなどの効果はない。

f-2:複数の printf 関数で出力してみる。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "      *\n" );
    printf( "     ***\n" );
    printf( "    *****\n" );
    printf( "     ***\n" );
    printf( "      *\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・複数の printf 関数でも、適度なスペースを挿入することで出力したい模様を出力できた。
・複数の printf 関数の間でも、改行によって次の行が一文字分ずれるなどの効果はない。
・視覚的に分かりやすく、調整しやすいのはこちらだと思われる。

g.「*」を用いて、自分の好きな形を出力せよ。

g:複数の printf 関数で出力する。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "                    *********\n" );
    printf( "                ******  ***  ******\n" );
    printf( "             ***        ***        ***\n" );
    printf( "           **           ***           **\n" );
    printf( "         **             ***             **\n" );
    printf( "        **              ***              **\n" );
    printf( "       **               ***               **\n" );
    printf( "       **               ***               **\n" );
    printf( "       **              *****              **\n" );
    printf( "       **            *********            **\n" );
    printf( "        **         *************         **\n" );
    printf( "         **      ****   ***   ****      **\n" );
    printf( "           **  ****     ***     ****  **\n" );
    printf( "             ****       ***       ****\n" );
    printf( "                ******  ***  ******\n" );
    printf( "                    *********\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・調整がしやすい f-2 項の方法でコードを入力した。

・意図した図形を出力することができた。

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。

h-1-A:"\b"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\b,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hell,C World!
```

考察▼

・メッセージ中に"\b"を挿入したところ、挿入した部分の直前、一文字が出力されない結果となった。

・次に、"\b"がメッセージ先頭にある場合と、"\n"直前にある場合、直後にある場合を試行する。

h-1-B:メッセージ先頭にある場合。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "\bHello,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hello,C World!
```

考察▼

・"\b"の直前にメッセージが存在しない場合、"\b"を入力しない場合と同じ結果が得られた。

h-1-C:"\n"の直前にある場合。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\b\n" );
    return(0);

}
```

実行結果▼

```
Hello,C World!
```

考察▼

・メッセージ中の"!"が出力されない結果になると予想していた。

・"\n"の直前に"\b"を挿入した場合も、"\b"を入力しない場合と同じ結果が得られた。

h-1-D:"\n"の直後にある場合。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\n\b" );
    return(0);

}
```

実行結果▼

```
Hello,C World!
```

考察▼

・改行効果が打ち消され、実行結果には"%"が出力されると予想していた。

・"\n"の直後に"\b"を挿入した場合も、"\b"を入力しない場合と同じ結果が得られた。

・次に、複数の printf 関数における"\b"の効果を確かめる。

h-1-E:複数のprintf関数の間における"\b"。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\n" );
    printf( "\bHello.C World!\n");
    return(0);

}
```

実行結果▼

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

考察▼

・1つ目のprintf関数での改行効果が打ち消され、一行で出力されると予想していた。
・こちらも"\b"を入力しない場合と同じ結果が得られた。
・h-1-B項からh-1-E項までの結果を踏まえ、"\b"の効果が表れることのできる適切な挿入位置を探す。

h-1-F:"\b"が効果を表す状況の模索。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\borld!\n" );
    return(0);

}
```

実行結果▼

```
Hello,C orld!
```

考察▼

・"\n"の前後や、メッセージ先頭を避け、適当な位置に挿入した。
・このとき、"\b"の直前の文字である"W"が出力されない結果が得られた。
・"\n"の前後やメッセージ先頭以外の箇所に"\b"を挿入する試行を繰り返した結果、全ての場合で"\b"の直前の文字（空白を含む）が出力されなかった。

・h-1-A 項から h-1-F 項までを踏まえると、"\b"は挿入位置が
メッセージの先頭や"\n"の前後でないとき、
直前の文字を出力しない効果を持つ。

h-2-A:"\f"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\f,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hello
     ,C World!
```

考察▼

・メッセージ中に"\f"を挿入したところ、上記のような実行結果が
得られた。

・次に、h-1-B 項から h-1-E 項までと同様に、
メッセージ先頭や"\n"の前後、複数の printf 関数の間においての
"\f"を観察する。

h-2-B:"\f"をメッセージ先頭に挿入する。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "\fHello,C World!\n" );
    return(0);

}
```

実行結果▼

```

Hello,C World!
```

考察▼

・h-2-A 項で得られた結果から、一行目は空白行となり、
二行目の一文字目から出力されると予想した。
・予想通りの結果が得られた。

h-2-C:"\f"を"\n"の直前に挿入する。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\f\n" );
    return(0);

}
```

実行結果▼　※二行目が空白行として出力された。

```
Hello,C World!

```

考察▼

・h-1-C 項（"\b"を"\n"の直前に挿入する試行）で得られた結果から、
"\f"を挿入しない場合と同じ結果が出力されると予想した。
・予想とは異なり、二行目が空白行として出力された。

h-2-D:"\f"を"\n"の直後に挿入する。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\n\f" );
    return(0);

}
```

実行結果▼　※二行目が空白行として出力された。

```
Hello,C World!

```

考察▼

・h-2-C 項（"\f"を"\n"の直前に挿入する試行）で得られた結果から、
一行目に"\f"を挿入しない場合と同じように出力され、
二行目は空白行として出力されると予想した。

・予想通り、"\n"の前後であっても"\f"を挿入しない場合とは
異なる結果が得られた。

h-2-E:複数の printf 関数の間における"\f"。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\n" );
    printf( "\fHello,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hello,C World!

Hello,C World!
```

考察▼

・h-2-C 項および h-2-D 項で得られた結果から、2 行目に空白行が
挿入されると予想した。
・予想通りの結果が得られた。
・最初の printf 関数のメッセージ最後尾に挿入した場合も同じ結果が
得られた。

h-2-F:"\f"が効果を表す状況の模索。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\forld!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・挿入出来うる場所について試行を繰り返した結果、
挿入地点よりあとの文字列を、縦に一行ずらすという変化が見られた。
・h-2-A 項から h-2-F 項までを踏まえると、"\f"は場所に関わらず、
挿入地点よりあとの文字列を、縦に一行ずらすという効果を持つ。
・縦にずらした地点まで足りない空白は自動で補われる。

h-3-A:"\n"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\n,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hello
,C World!
```

考察▼

・今まで改行として扱ってきた"\n"を改めて観察する。
・適当な位置に挿入した結果、その直後で改行された。

h-3-B:複数の printf 関数の間における"\n"。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\n\n" );
    printf( "Hello,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hello,C World!

Hello,C World!
```

考察▼

・最初の printf 関数のメッセージ部最後尾に"\n"を 2 つ入力した結果、
上記のような結果が得られた。
・2 つ目の printf 関数のメッセージ部先頭に"\n"を入力しても
同じ結果が得られた。
・同じ結果であっても、その空白行がどちらの printf 関数によって
出力されているかは別であると考えられる。

h-3-C:"\n"が効果を表す状況の模索。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\norld!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・メッセージ内において、挿入出来うる場所への試行を繰り返した結果、
挿入した場所から後の文字列を、"\f"とは違い空白に置き換えずに、
先頭に詰めて改行する変化が見られた。
・"\n"が 2 回続いた場合、上の変化に加え空白行が 1 つ追加される。
・このことから、"\n"は場所に関わらず、
挿入された場所からあとの文に対し改行する効果を持つ。

h-4-A:"\r"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\r,C World!\n" );
    return(0);

}
```

実行結果▼

```
,C World!
```

考察▼

・メッセージ中の適当な場所に"\r"を挿入した結果、挿入地点から
前の文字列が全て出力されないという結果が得られた。

・次に、メッセージ部先頭や、"\n"の前後、メッセージ内の挿入出来うる
場所における"\r"の効果を観察する。

h-4-B:メッセージ部先頭、"\n"前後における"\r"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "\rHello,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hello,C World!
```

考察▼

・メッセージ部先頭に"\r"を挿入した結果、"\r"を挿入しないときと同じ
結果が得られた。

・"\n"の前後においても試行したときも同様に、"\r"を挿入しないときと
同じ結果が得られた。

h-4-C:複数の printf 関数の間における"\r"。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\n\r" );
    printf( "Hello,C World!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・最初の printf 関数のメッセージ部最後尾に"\r"を挿入した結果、
"\r"を挿入しないときと同じ結果が得られた。

・2 つ目の printf 関数のメッセージ部先頭に"\r"を挿入しても
同様の結果が得られた。

h-4-D:"\r"が効果を表す状況の模索。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\rorld!\n" );
    return(0);

}
```

実行結果▼

```
orld!,C W
```

考察▼

・"\n"の前後や、メッセージ先頭を避け、適当な位置に挿入した結果、
"\r"より前にある文字列が出力されなかった。

・"\n"の前後やメッセージ先頭以外の箇所に"\r"を挿入する試行を
繰り返した。

・結果、特定の地点よりあとの地点に"\r"を挿入したとき、
"\r"のあとの文が通常通り出力されたあと
"\r"より前の部分の数文字が出力されるという結果が得られた。
・メッセージの内容を変えて試行し、変化を観察する。

h-4-E:"\r"の挙動の観察。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "AAAAaaaaBBBB\rbbbbCCCCcccc\n" );
    return(0);

}
```

実行結果▼

```
bbbbCCCCcccc
```

考察▼

・"\n"を除いて、"\r"の前後でメッセージの文字数が等しくなるように
"\r"を挿入した。
・結果、"\r"より前の文字列は出力されなかった。
・"\r"の挿入位置を四文字分右にずらし、もう一度観察する。

h-4-F:"\r"の挙動の観察 2。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "AAAAaaaaBBBBbbbb\rCCCCcccc\n" );
    return(0);

}
```

実行結果▼

```
CCCCccccBBBBbbbb
```

考察▼

・"\r"の挿入位置を四文字分右にずらした結果、"CCCCcccc"の
文字列のあとに"BBBBbbbb"という文字列が出力された。

・コード上ではそのような入力はしていないので、
これが"\r"の効果であると考えられる。
・更に右に四文字分ずらし、観察する。

h-4-G:"\r"の挙動の観察 3。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "AAAAaaaaBBBBbbbbCCCC\rcccc\n" );
    return(0);

}
```

実行結果▼

```
ccccaaaaBBBBbbbbCCCC
```

考察▼

・"\r"の挿入位置をさらに四文字分右にずらすと、上記のような結果が
得られた。
・h-4-D 項から h-4-G 項までを踏まえると、"\r"が挿入された地点より
後の文字列を、行の先頭に上書きしていくような変化が見られる。
・メッセージの文字数の半分を超えていない地点で挿入された場合、
"\r"より前を全て消去したかのような挙動に見えてしまう。
・メッセージの文字数の半分を超えていれば、"\r"より後の文字列で
上書きがされなかった分の文字列が出力される。

h-5-A:"\t"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\t,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hello    ,C World!
```

考察▼

・メッセージ中の適当な場所に"\t"を挿入した結果、
　挿入地点に空白が表れた。
・次に、メッセージ部先頭に挿入した場合の挙動を観察する。
・何文字分の空白が出力されているのかを調べる為、
　2 つ目の printf 関数を用意する。

h-5-B:メッセージ部先頭における"\t"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "\tHello,C World!\n" );
    printf( "AAAAAAAAAAAAAAAAAAAA\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・メッセージ部先頭に挿入した場合、大きな空白が出力された。
・出力された空白の幅は半角八文字分。
・h-5-A 項で出力された空白とは幅が異なる。
・挿入場所を適当に変えて観察する。

h-5-C:メッセージ先頭から八文字を超えない部分における\t"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C\t World!\n" );
    printf( "AAAAAAAAAAAAAAAAAAAA\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・メッセージ部中間に挿入した場合、出力された空白は小さな物だった。
・出力された空白の幅は、半角一文字分。
・メッセージ中の空白のうち、
　一文字分はメッセージとして出力されているものである。
・h-5-B 項、h-5-C 項より、メッセージ先頭から七文字目までに"\r"が
　挿入されると、その直後の文字列が八文字目から出力されるように
　空白が出力されているように見える。

h-5-D:メッセージ先頭から八文字を超えた部分における"\t"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C\t World!\n" );
    printf( "AAAAAAAAAAAAAAAAAAAA\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・先頭から八文字を超えた場合、新たに八文字分の空白が出力された。
・次に、挿入位置を四文字分右にずらして観察する。

h-5-E:更に四文字分ずらした場所における"\t"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C Worl\td!\n" );
    printf( "AAAAAAAAAAAAAAAAAAAA\n" );
    return(0);

}
```

実行結果▼

```
Hello,C Worl    d!
AAAAAAAAAAAAAAAAAAAA
```

考察▼

・挿入位置を四文字分右にずらした結果、出力される空白の幅が
四文字分減った。

・h-5-B 項から h-5-E 項の結果を踏まえると、
プログラムには半角八文字分ごとに特殊な地点があると考えられる。
"\t"は、挿入された地点からあとの文字列を、次の「特殊な地点」まで
空白を補って移動させる効果がある。

h-5-F:複数の printf 関数の間における"\t"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\torld!\n" );
    printf( "Hello,C World!\n" );
    return(0);

}
```

実行結果▼

```
Hello,C W    orld!
Hello,C World!
```

考察▼

・複数の printf 関数の間においても、"\t"は h-5-E 項で示したような
効果を表す。

・異なる行の文字列には影響を与えない。

h-6-A:"\'"（\アポストロフィー）について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\',C World!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・メッセージ中に"\'"を挿入したところ、挿入した地点に
アポストロフィー（'）が出力された。

・メッセージ部先頭や"\n"の前後に挿入し、挙動を観察する。

h-6-B:メッセージ部先頭や"\n"の前後における"\'"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "\'Hello,C World!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・h-6-A項より、メッセージ先頭にアポストロフィーが
出力されると予想した。

・メッセージ部先頭に"\'"を挿入すると、予想通りの結果が得られた。

・"\n"の前後においても同様の試行を行ったが、いずれの場合も
アポストロフィーが出力された。

h-6-C:メッセージ部先頭や"\n"の前後における"\'"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\'orld!\n" );
    printf( "Hello,C W\'orld!\n" );
    return(0);

}
```

実行結果▼

```
Hello,C W'orld!
Hello,C W'orld!
```

考察▼

・複数の printf 関数においても、"\'"に挿入した地点に
アポストロフィーが出力される。
・h-6-A 項から h-6-C 項までを踏まえると、
"\'"は挿入した地点にアポストロフィーを出力する効果を持つ。
・ただし、バックスラッシュを用いずメッセージ部に
アポストロフィーだけを挿入した場合であってもアポストロフィーが
出力された。

h-7-A:"\""（\ダブルクォーテーション）について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\",C World!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・メッセージ中に"\""を挿入したところ、挿入した地点に
ダブルクォーテーション（"）が出力された。
・メッセージ部先頭や"\n"の前後に挿入し、挙動を観察する。

h-7-B:メッセージ部先頭や"\n"の前後における"\""について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "\"Hello,C World!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・h-7-A 項より、メッセージ先頭にダブルクォーテーションが
出力されると予想した。
・メッセージ部先頭に"\""を挿入すると、予想通りの結果が得られた。
・"\n"の前後においても同様の試行を行ったが、いずれの場合も
ダブルクォーテーションが出力された。
・a-2 項では出力できなかったことから、特殊な文字である
ダブルクォーテーションを文字として出力する効果であると考えられる。

h-7-C:メッセージ部先頭や"\n"の前後における"\'"について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\"orld!\n" );
    printf( "Hello,C W\"orld!\n" );
    return(0);

}
```

実行結果▼

```
Hello,C W"orld!
Hello,C W"orld!
```

考察▼

・複数の printf 関数においても、h-6-B 項で予想した効果が表れている。
・他の行に影響を与えることはない。

h-8-A:"\\"（\バックスラッシュ）について。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\\,C World!\n" );
    return(0);

}
```

実行結果▼

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

考察▼

・メッセージ中に"\\"を挿入したところ、挿入した地点に
バックスラッシュ（\）が出力された。

・h-7 項や h-8 項と同様に、
メッセージ部先頭や"\n"の前後への挿入、
複数の printf 関数の間における挙動を観察した結果、
いずれも挿入地点にバックスラッシュが出力された。

・よって、"\\"は挿入した地点にバックスラッシュを出力する効果を持つと考えられる。

・a-2 項では出力できなかったことから、特殊な文字である
バックスラッシュを文字として出力する効果であると考えられる。

・他の行には影響を与えない。

h-9-A:"\nnn"（nnn の部分には 8 進数の数字を入力する）について。
（\000～\012 編）

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\\000\n" );
    printf( "Hello\001,C World!\\001\n" );
    printf( "Hello\002,C World!\\002\n" );
    printf( "Hello\003,C World!\\003\n" );
    printf( "Hello\004,C World!\\004\n" );
    printf( "Hello\005,C World!\\005\n" );
    printf( "Hello\006,C World!\\006\n" );
    printf( "Hello\007,C World!\\007\n" );
    printf( "Hello\010,C World!\\010\n" );
    printf( "Hello\011,C World!\\011\n" );
    printf( "Hello\012,C World!\\012\n" );
    return(0);

}
```

コンパイルの際に出たエラー▼

```
YP9.c: In function ‘main’:
YP9.c:13: warning: embedded ‘\0’ in format
YP9.c:13: warning: embedded ‘\0’ in format
```

考察▼

・テキスト PP407~408 を参考にすると、"0"から"32"まで
（8 進数で"000"から"040"）の数字には、通常の文字ではなく
特殊な効果が割り振られているようだ。

・"\000"から"\012"までを一斉に試行しようとしたところ、
上記のようなエラーが表れた。

・"\0"が含まれていることが原因のようなので、"\000"をコードから
消去して再試行する。

h-9-B:"\nnn"について。（nnn の部分には 8 進数の数字を入力する）
（\001～\012 編）

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C World!\\000\n" );
    printf( "Hello\001,C World!\\001\n" );
    printf( "Hello\002,C World!\\002\n" );
    printf( "Hello\003,C World!\\003\n" );
    printf( "Hello\004,C World!\\004\n" );
    printf( "Hello\005,C World!\\005\n" );
    printf( "Hello\006,C World!\\006\n" );
    printf( "Hello\007,C World!\\007\n" );
    printf( "Hello\010,C World!\\010\n" );
    printf( "Hello\011,C World!\\011\n" );
    printf( "Hello\012,C World!\\012\n" );
    return(0);

}
```

実行結果▼

```
Hello,C World!\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
Hell,C World!\010
Hello   ,C World!\011
Hello
,C World!\012
```

考察▼

・"\000"をコードから消去することでコンパイルをすることができた。
・上記の結果より、"\001"から"\007"までは効果が表れていないように考えられる。
・"\010"では、h-1 項での"\b"と同様に挿入位置直前の文字が出力されていない。
・"\011"では、"\t"を入力したときのような空白が出力された。
・"\012"では、"\n"を入力したときのように改行されている。
・よって、"\010"は"\b"、"\011"は"\t"、"\012"は"\n"相当の効果を持っていると考えられる。

h-9-B+:"\010"、"\011"、"\012"の効果の確認。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\010orld!\\010\n" );
    printf( "Hello,C \011World!\\011\n" );
    printf( "Hello,C World!\012\\012\n" );
    return(0);

}
```

実行結果▼

```
Hello,C orld!\010
Hello,C  World!\011
Hello,C World!
\012
```

考察▼

・"\010"について、やはり"\b"と同様に挿入地点の直前の一文字を出力しない効果があるようだ。

・"\011"、"\012"についても同様に、h-9-B 項で予想した通りそれぞれ"\t"や"\n"に対応する効果が表れた。

h-9-C:"\nnn"（nnn の部分には 8 進数の数字を入力する）について。

（\013～\025 編）

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\013,C World!\\013\n" );
    printf( "Hello\014,C World!\\014\n" );
    printf( "Hello\015,C World!\\015\n" );
    printf( "Hello\016,C World!\\016\n" );
    printf( "Hello\017,C World!\\017\n" );
    printf( "Hello\020,C World!\\020\n" );
    printf( "Hello\021,C World!\\021\n" );
    printf( "Hello\022,C World!\\022\n" );
    printf( "Hello\023,C World!\\023\n" );
    printf( "Hello\024,C World!\\024\n" );
    printf( "Hello\025,C World!\\025\n" );
    return(0);

}
```

実行結果▼

```
Hello
     ,C World!\013
Hello
     ,C World!\014
,C World!\015
Hello,C World!\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
```

考察▼

・"\013"から"\025"までを入力した。

・上記の結果より、"\016"から"\025"までは効果が表れていないように考えられる。

・"\013"、"\014"については、"\f"と同様な結果が表れている。

・"\015"については、"\r"に似た結果が表れている。

h-9-C+:"\013"、"\014"、"\015"の効果の確認。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C W\013orld!\\013\n" );
    printf( "Hello,C \014World!\\014\n" );
    printf( "aaaaAAAAbbbbBBBB\015ccccCCCC\\015\n" );
    return(0);

}
```

実行結果▼

```
Hello,C W
         orld!\013
Hello,C
        World!\014
ccccCCCC\015BBBB
```

考察▼

・"\013"、"\014"について、"\f"と同様に
挿入地点よりあとの文字列を縦に一行ずらす効果が確認できた。

・"\015"についても、"\r"と同様の効果が確認できた。

h-9-D:"\nnn"（nnn の部分には 8 進数の数字を入力する）について。
（\026～\040 編）

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\026,C World!\\026\n" );
    printf( "Hello\027,C World!\\027\n" );
    printf( "Hello\030,C World!\\030\n" );
    printf( "Hello\031,C World!\\031\n" );
    printf( "Hello\032,C World!\\032\n" );
    printf( "Hello\033,C World!\\033\n" );
    printf( "Hello\034,C World!\\034\n" );
    printf( "Hello\035,C World!\\035\n" );
    printf( "Hello\036,C World!\\036\n" );
    printf( "Hello\037,C World!\\037\n" );
    printf( "Hello\040,C World!\\040\n" );
    return(0);

}
```

実行結果▼

```
Hello,C World!\026
Hello,C World!\027
Hello,C World!\030
Hello,C World!\031
Hello,C World!\032
HelloC World!\033
Hello,C World!\034
Hello,C World!\035
Hello,C World!\036
Hello,C World!\037
Hello ,C World!\040
```

考察▼

・"\026"から"\040"までを入力した。
・上記の結果より、"\026"から"\032"、"\034"から"\037"は、
効果が表れていないように考えられる。
・"\033"について、挿入した地点の直後の文字が出力されていない。
・"\040"について、半角一文字分の空白が出力されている。
・"\033"、"\040"について挙動を観察する。

h-9-D+:"\033"、"\040"の効果の確認。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello,C \033World!\\013\n" );
    printf( "Hello,C W\040orld!\\014\n" );
    return(0);

}
```

実行結果▼

```
Hello,C orld!\013
Hello,C W orld!\014
```

考察▼

・"\033"について、やはり挿入した地点の直後の一文字が
出力されていない。
・よって、"\033"は入力された地点の直後にある文字を一文字だけ
出力しないという効果であると考えられる。
・"\040"についても、h-9-D 項での結果と同じく
入力した地点に半角一文字分の空白が出力されている。
・したがって、"\040"は半角スペースを一文字分出力する効果であると
考えられる。

h-9-E:"\nnn"（nnn の部分には 8 進数の数字を入力する）について。
（\041 以降編）

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "Hello\041,C World!\\041\n" );
    return(0);

}
```

実行結果▼

```
Hello!,C World!\041
```

考察▼

・"\041"以降について観察していく。
・"\041"では、入力した地点に"!"が出力された。
・"\042"、"\043"、"\044".....と試行を繰り返した結果、
"\041"以降の"\nnn"は全て特定の一文字を出力することが
確認できた。
・ただし"\045"はコンパイルの際にエラーが表示され、
"\177"は変化が確認できなかった。

"\045"を入力し、コンパイルしようとした際のエラー▼

```
YP9X.c: In function ‘main’:
YP9X.c:13: warning: unknown conversion type character ‘,’ in format
YP9X.c:13: warning: unknown conversion type character ‘,’ in format
```

エラーの考察▼

・"\045"が入力された地点の直後の文字がエラーの対称となっている。
・"\045"だけの文であっても、下記のようなエラーが表示され
コンパイルできない。

"\045"のみを入力し、コンパイルしようとした際のエラー▼

```
YP9X.c: In function ‘main’:
YP9X.c:13: warning: spurious trailing ‘%’ in format
YP9X.c:13: warning: spurious trailing ‘%’ in format
```

エラーの考察▼

・エラー文から、"\045"は"%"を出力する効果があるのかもしれない。
・a-2 項（いろいろな文字種の試行）でのエラーと共に、
次項で考察する。

i.エラーについて考察せよ。

i:"%"について。

推測▼

・a-2 項では"%"、"!"や"?"、"^"、"\"、")"、"@"、";"、":"、
"ダブルクォーテーション(")"が出力できないとしたが、
他の項目では"!"を出力できていることから他に原因があると推測した。
・a-2 項において最後に消去したのは"%"であり、
h-9-E 項において"\045"を入力した際に"%"に関してのエラーが
表れることから、"%"がエラーの原因ではないか。
・"\"や"ダブルクォーテーション(")"については、プログラムの表記に
必要な文字であることから出力できないことが通常であるとし、
出力できないとした上記から"%"、"\"、"ダブルクォーテーション"
を消去し出力を試みる。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "! ? ^ ) @ ; :\n" );
    return(0);

}
```

実行結果▼

```
! ? ^ ) @ ; :
```

考察▼

・出力できないとした文字列を出力することができた。
・"%"がメッセージ中に含まれているとコンパイルの段階で
エラーが表れる。
・どうすれば"%"を出力できるのかを考える。

高水準コードの一部▼

```
#include <stdio.h>

int main(){

    printf( "%%\n" );
    return(0);

}
```

実行結果▼

```
%
```

考察▼

・"\"が"\\"で出力できることから、二回連続で入力すれば
出力できると期待した。
・結果の通り、"%"を出力することができた。
・同じく"%"に関連しているであろう"\045"に対しても
同様の試行を行うと、こちらも"%"を出力することができた。
・これらの結果から、
"%"および"\045"は、二回連続で入力することで
printf 関数で出力できることが確認できた。

XXX.あとがき（反省・感想）

気になったことを次から次へと試しているとページ数が膨大なものになり、
本当にこれで良いのかと書いた自分自身が一番怯えています。
次回のレポートでは、気になったことを考察し、その上で
「効率良く」レポートを書くように善処したいです。
効率に気を奪われて、内容が劣るといった事態にならないよう注意します。

ページ数といえば、これだけのページ数を印刷するのは凄く大変なことだと
印刷する前から予想してしまっているのですが、何故プログラミングの
レポートは紙媒体で提出するのでしょうか。
pdf での提出にすれば、紙やインク（トナー、でしょうか）の節約に繋がり
また谷口先生のオフィスに 65 人分の紙束が押し寄せることもなくなると
思います。
純粋な疑問ですので、気を悪くしないでください。
最後までご覧いただき、ありがとうございました。