1 asir から C へ. 例題プログラム集

1.1 関数定義

```
def nsum(N) {
  S=0;
  for (I=1; I<=N; I++) {
    S = S+I;
  }
  return(S);
}
def main() {
  n=nsum(100);
  printf("%a\n",n);
}
main();
```

```
#include <stdio.h>
int nsum(int n) {
  int i,s;
  s=0;
  for (i=1; i<=n; i++) {
    s = s+i;
  }
  return(s);
}
main() {
  int s;
  s=nsum(100);
  printf("%d\n",s);
}
```

1.2 配列

```
def main() {
  S = newvect(4);
  S[0] = 0x41;
  S[1] = 0x42;
  S[2] = 0x43;
  S[3] = 0xa;
  S2 = asciitostr(vtol(S));
  printf("%a\n",S2);
}
main();
end$
```

```
#include <stdio.h>
main() {
  unsigned char s[10];
  s[0] = 0x41;
  s[1] = 0x42;
  s[2] = 0x43;
  s[3] = 0xa;
  s[4] = 0;
  printf("%s\n",s);
}
```

1.3 ファイル入出力

ファイルとは 1 byte の数の列である. ファイルから 1 byte のデータを読み取り, 16 進数として表示するプログラムは以下のとおり.

```
def dump(FileName) {
  Fp = open_file(FileName);
  if (Fp < 0) error("Open failed.");
  while ((C=get_byte(Fp)) >= 0) {
    printf("%a, ",C);
  }
}
dump("dump.c");
end$
```

```
#include <stdio.h>
main() {
  FILE *fp;
  int c;
  fp = fopen("dump.c","r");
  if (fp == NULL) {
    printf("Open error.\n");
    return(-1);
  }
  while ((c=fgetc(fp)) >= 0) {
    printf("%3x ",c);
  }
  fclose(fp);
}
```

1.4 2 の補数表現

```
#include <stdio.h>
main() {
  unsigned char a,b;
  a = 0x7f;
  b = 0x81;
  printf("%d\n",(int) (a+b));
}
```

Asir と異なり C 言語では, `unsigned char` 型の数の計算は, mod 0x100 での計算 (1 byte), `unsinged int` 型の数の計算は, 通常の処理系では, mod 0x100000000 (4 byte) での計算となる. 0x7f + 0x81 = 0x100 であるが, 0x100 で割ると余りが 0 となるので, 0 が左のプログラムでは表示される.
問. `b=0x82` と変更するといくつが表示されるか?

参考: int 型の変数は通常の処理系では 4 byte の領域を占める. mod 0x100000000 の剰余計算で四則計算が処理されるが, 負の数を表すために, 0x7fffffff を最大の正の数とし, 0xffffffff を -1, 0xfffffffe を -2, 0xfffffffd を -3 と解釈する. この方法を用いると, mod 計算で負の数を含む計算が正しく処理されているように見える. たとえば, 0xffffffff + 0x00000002 = 0x00000001 mod 0x100000000 であるが, たしかに $-1+2=1$ である. 0xffffffff = 0-1 mod 0x100000000 なる関係式を用いて考えれば, 掛け算を含む計算が桁が小さい時には普通の四則計算に見えることを示すことができる.

参考プログラム.

```
#include <stdio.h>
main() {
  unsigned int a,b;
  int x,y;
  a = 0x7f000000;
  b = 0x81000000;
  printf("%d\n",a+b);
  x = 0x7f000000;
  y = 0x81000000;
  printf("%d\n",a+b);
  printf("----------------------\n");
  a = 0x7f000000;
  b = 0x80fffffe;
  printf("%ld\n",(long int) a+b);
  printf("%x\n",a+b);
  x = 0x7f000000;
  y = 0x80fffffe;
  printf("%d\n",x+y);
  printf("%x\n",x+y);
}
```

出力は

```
0
0
----------------------
4294967294
fffffffe
-2
fffffffe
```

%x は 16 進数を用いて内部的な表現を表示せよ、という意味.