ETH Zürich
Departement ITET
Institut TIK
Prof. Dr. Thomas Erlebach

WS 2003/2004
ACM Programming Contest
Information 2
29. Oktober 2003

PPS-Praktikum

ACM Programming Contest

Registrierung auf dem ACM-Server

Um sich als Mitglied auf dem ACM Problemset Webserver zu registrieren, gehen Sie bitte auf die Webseite

http://acm.uva.es/problemset/

und klicken Sie unter "Users Manager" den Punkt "Register as a new member" an. Füllen Sie die daraufhin angezeigten Web-Formulare aus. Dabei können Sie ein beliebiges Passwort wählen und müssen dann Ihre persönlichen Daten (Name, Email-Adresse, etc.) eingeben. Am Ende erhalten Sie eine Benutzernummer der Form "1000AA". Senden Sie Ihre Benutzernummer bitte auch per Email an Thomas Erlebach (erlebach@tik.ee.ethz.ch).

Die Programme, die Sie (als Quelltext!) an den Online-Judge schicken, müssen mit einem Kommentar der folgenden Form beginnen:

```
/* @JUDGE_ID: 1000AA 166 C++ "easy job" */
```

Dabei ist "1000AA" Ihre Benutzernummer, "166" die Nummer des Problems, "C++" die Programmiersprache (es geht auch "C", "Pascal", oder "Java") und "easy job" ein optionaler Kommentar, der dann ggf. im Ranking auf der ACM-Webseite erscheint.

Stellen Sie sicher, dass Sie in Ihrer Rechnerumgebung C- oder C++-Programme (ANSI-Standard!) kompilieren und ausführen können. Ein C-Programm "x.c" und ein C++-Programm "x.C" sollten Sie unter Unix mit gcc/g++ zum Beispiel mit den folgenden Kommandos übersetzen können:

```
gcc -ansi -o x x.c -lm
g++ -ansi -o x x.C -lm
```

Um ein Programm an den Judge zu schicken, verwenden Sie ein Kommando der folgenden Art:

```
mail judge@uva.es <x.c
```

Wenn Sie Lösungen für die wöchentlichen Programmieraufgaben oder die Semesteraufgaben abgeben, schicken Sie bitte **den Quelltext und die Antwort des Judge** an Thomas Erlebach (erlebach@tik.ee.ethz.ch).

Ein- und Ausgabe in C und C++

Die Programme zur Lösung der ACM-Programmieraufgaben lesen ihre Eingabe von `stdin` (Standardeingabe, in C++ `cin`) und schreiben die Ausgabe nach `stdout` (Standardausgabe, in C++ `cout`). Hier einige Beispiele.

```
/* Aufsummierung von Zahlen in der Eingabe jeweils bis zur nächsten
Null, Programmende bei Erreichen von -1 */
#include <stdio.h>
int main(){
  int d,r=0;
  for (;;) {
    scanf("%d",&d);
    if (d==-1) break;
    else if (d==0) { printf("%d\n",r); r=0; }
    else r+=d;
  }
  return 0;
}
```

Dasselbe in C++:

```
#include <iostream.h>
int main(){
  int d,r=0;
  for (;;) {
    cin >>d;
    if (d==-1) break;
    else if (d==0) { cout <<r <<endl; r=0; }
    else r+=d;
  }
  return 0;
}
```

Die folgenden Programme lesen Zahlen aus der Eingabe, bis das Dateiende erreicht ist, und geben dann die Summe aus:

```
#include <stdio.h>
int main(){
  int d,r=0;
  while (scanf("%d",&d)==1) r+=d;
  printf("%d\n",r);
  return 0; }
```

In C++:

```
#include <iostream.h>
int main(){
  int d,r=0;
  while (cin>>d) r+=d;
  cout <<r <<'\n';
  return 0; }
```

Bei den bisherigen Programmen wurden Zeilenende-Zeichen in der Eingabe wie Leerzeichen behandelt. Wenn die Eingabe zeilenweise behandelt werden soll, kann man wie folgt vorgehen:

```
/* Aufsummierung der Zahlen in jeder Eingabezeile und Ausgabe
des Ergebnisses jeder Zeile, Ende bei Leerzeile */
#include <stdio.h>
#include <stdlib.h>
int main(){
  int r;
  char buf[1024]; /* Zeilenbuffer fuer bis zu 1024 Zeichen */
  char *p;
  for (;;) {
    fgets(buf,sizeof(buf),stdin); /* liest '\n' am Zeilenende mit!! */
    if (buf[0]=='\n') break; /* Leerzeile gelesen */
    p = buf; /* Zeiger auf Buffer-Anfang */
    r=0;
    do {
      r += strtol(p,&p,10); /* naechste Zahl kovertieren und addieren */
    } while (*p!='\n');
    printf("%d\n",r);
  }
  return 0;
}
```

Wieder dasselbe in C++:

```
#include <iostream.h>
#include <stdlib.h>
int main(){
  int r;
  char buf[1024]; /* Zeilenbuffer fuer bis zu 1024 Zeichen */
  char *p;
  for (;;) {
    cin.getline(buf,sizeof(buf)); /* '\n' wird nicht mitgespeichert! */
    if (buf[0]==0) break; /* Leerzeile gelesen */
    p = buf; /* Zeiger auf Buffer-Anfang */
    r=0;
    do {
      r += strtol(p,&p,10); /* naechste Zahl kovertieren und addieren */
    } while (*p);
    cout <<r <<endl;
  }
  return 0;
}
```

Schliesslich noch ein Beispiel, wie man bei Aufgaben mit sehr grosser Eingabe/Ausgabe durch Pufferung die Laufzeit stark beschleunigen kann:

```
/* liest Zahlen aus der Eingabe und gibt für je zwei aufeinander-
folgende Zahlen die Summe aus; Terminierung bei Lesen von 0 0 */
#include <stdio.h>
#define BUFSIZE 8192
char inbuf[BUFSIZE];
char outbuf[BUFSIZE];
int insize=0; /* Zeichen im inbuffer */
int inp=0; /* Index des aktuellen Zeichens im inbuffer */
int outp=0; /* Index in Ausgabepuffer */

int getnum() { /* gepuffertes Lesen einer Zahl */
  int rest, s;
  char *ende;
  rest = insize-inp; /* restliche Zeichen im Puffer */
  if (rest<10 && !feof(stdin)) { /* naechsten Block lesen */
     memcpy(inbuf,inbuf+inp,rest); /* Rest nach vorne kopieren */
     insize = rest + fread(inbuf+rest,1,BUFSIZE-rest,stdin);
     inp=0; /* reset Index */
  }
  s = strtol(inbuf+inp,&ende,10);
  inp = ende - inbuf;
  return s;
}

int putnum(int s) { /* gepuffertes Schreiben der Zahl s */
  if (outp>BUFSIZE-10) { /* kein Platz mehr */
     fwrite (outbuf,1,outp,stdout); /* outbuf ausgeben */
     outp=0;
  }
  sprintf(&outbuf[outp],"%d\n",s);
  outp+=strlen(&outbuf[outp]); /* outp aktualisieren */
}

main() {
  int u,v;
  insize = fread(inbuf,1,BUFSIZE,stdin); /* Ersten Block lesen */
  for (;;) {
    u = getnum();
    v = getnum();
    if (u==0 && v==0) break;
    putnum(u+v);
  }
  if (outp>0)
     fwrite (outbuf,1,outp,stdout); /* Rest von outbuf ausgeben */
}
```