Matthias Jauernig
Kurs: 03In1

02.12.03

8. Aufgabenserie zu den Grundlagen der Informatik

Abgabetermin: Mi, 03.12.03

Zu 22.) *Infix-, Präfix- und Postfix-Notation* ganzzahliger Terme

Infix-Notation	Präfix-Notation	Postfix-Notation
(((i+j)%4) – (s<<t) / (p | abs(q)))	**- % + i j 4 / << s t | p abs q**	**i j + 4 % s t << p q abs | / -**
(((j&k) / i) % j)	% / & j k i j	**j k & i / j %**
~(r*(((s*t) | i) + j))	**~ * r + | * s t i j**	r s t * i | j + * ~

Zu 23.) *Potenzmenge* von {1,…,n}

→ Zu dieser Aufgabe habe ich drei verschiedene Programme, die folgend aufgeführt sind:

- potmenge_kl.c gibt bis n=31 ungeordnet aus
- potmenge_gr.c gibt bis n=32767 aus, auch wenn dies wohl nie jemand machen würde :o)
- potmenge_geordn.c gibt bis n=24 geordnet aus; der kleine Wert für n kommt durch den hohen Speicherbedarf des zusätzlichen Arrays a zustande (siehe Kommentare im Programm: ca. 45MByte Bedarf!!!) und kann daher nicht größer gewählt werden

---

1.) potmenge_kl.c

```
/*      potmenge_kl.c -- Matthias Jauernig, 01.12.03                    */
/*      Programm gibt die Potenzmenge einer natürlichen Zahl n>=1 aus,  */
/*      also alle 2^n Teilmengen der Menge {1,...,n}  -- ungeordnete Ausgabe */

#include <stdio.h>
// BITS ist eine symbolische Konstante für die max. Anzahl von n (=Bits in Dual)
#define BITS 32

// ausg_menge() - gibt die momentane Menge anhand der Dualzahl aus
void ausg_menge(long int dual, int n){
        int i;
        printf("{");
        // gib den Index der Dualziffern aus, die in der Zahl "1" sind
        for(i=1; i<=n; i++)
                if((dual>>(n-i))&1)
                        printf("%d,", i);
        printf("\b},");
}

// pow() - gibt die Zahl basis^exp zurück und ist Ersatz für pow() aus der math.h
long int pow(int basis, int exp){
        int pot=1,i;
        for(i=1; i<=exp; i++)
                pot*=basis;
        return pot;
}
```

```c
// main() ------------------------------------------------------------------------
int main(void){
	int n, i;

	printf( "-----------------------------\n"
		"| Potenzmenge von {1,...,n} |\n"
		"-----------------------------\n\n");
	do{
		printf("Geben Sie eine ganze Zahl n>=1, n<%d ein: ", BITS);
		scanf("%d", &n);
	}while((n<1 && printf("Zahl muss größer oder gleich 1 sein!\n\n"))
		|| (n>BITS-1 && printf("Zahl muss kleiner %d sein!\n\n", BITS)));

	printf("\nP({1,...,%d})={ {},", n);
	// gib alle möglichen 2^n Kombinationen aus
	for(i=1; i<=pow(2,n)-1; i++)
		ausg_menge(i, n);
	printf("\b }\n\n");

	return 0;
}

/* Beispielausgabe:
-----------------------------
| Potenzmenge von {1,...,n} |
-----------------------------

Geben Sie eine ganze Zahl n>=1, n<32 ein: 6

P({1,...,6})={
{},{6},{5},{5,6},{4},{4,6},{4,5},{4,5,6},{3},{3,6},{3,5},{3,5,6},{3,4},{3,4,6},
{3,4,5},{3,4,5,6},{2},{2,6},{2,5},{2,5,6},{2,4},{2,4,6},{2,4,5},{2,4,5,6},{2,3},
{2,3,6},{2,3,5},{2,3,5,6},{2,3,4},{2,3,4,6},{2,3,4,5},{2,3,4,5,6},{1},{1,6},{1,5},
{1,5,6},{1,4},{1,4,6},{1,4,5},{1,4,5,6},{1,3},{1,3,6},{1,3,5},{1,3,5,6},{1,3,4},
{1,3,4,6},{1,3,4,5},{1,3,4,5,6},{1,2},{1,2,6},{1,2,5},{1,2,5,6},{1,2,4},{1,2,4,6},
{1,2,4,5},{1,2,4,5,6},{1,2,3},{1,2,3,6},{1,2,3,5},{1,2,3,5,6},{1,2,3,4},{1,2,3,4,6},
{1,2,3,4,5},{1,2,3,4,5,6} } */
```

2.) potmenge_gr.c

```c
/*	potmenge_gr.c -- Matthias Jauernig, 02.12.03				*/
/*	Programm kann bis n=32767 die Potenzmengen ausgeben, auch wenn dies	*/
/*	sicher einige Wochen Zeit in Anspruch nehmen würde ;o)			*/
/*	Programm gibt ungeordnet aus!						*/

#include <stdio.h>
#include <stdlib.h>
//maximales n festlegen
#define ELEM_MAX 32767

//add_dual() -- addiert eine 1 zu dem String Elem und gibt den Überlauf zurück
short add_dual(char *Elem, short n){
	int i, ende=0;
	//addiere dual 1 zu dem "Binärzahlen"-String
	for(i=0; !ende; i++){
		if(i==n) ende=1;
		if(Elem[i]=='0'){
			Elem[i]='1';
			ende=1;
		}
		else	Elem[i]='0';
	}
```

```
        //es gibt einen Überlauf oder nicht
        if(Elem[n]=='1')
                return 1;
        return 0;
}


int main(void){
        char Elem[ELEM_MAX];
        short n, i;

        printf( "-----------------------------\n"
                "| Potenzmenge von {1,...,n} |\n"
                "-----------------------------\n\n");
        do{
                printf("Geben Sie eine Zahl n ein: ");
                scanf("%hd",&n);
        }while(n<1 && printf("n muss groesser als 0 sein!\n\n"));

        // setze zunächst alle chars "0"
        for(i=0; i<ELEM_MAX; i++)
                Elem[i]='0';

        printf("\nP={ {},");
        //Schleife läuft, bis add_dual()==1, also alle Potenzmengen ausgegeben wurden
        while(!add_dual(Elem, n)){
                printf("{");
                //betrachte Dualziffern der Reihe nach, gib nach Bedarf den Index aus
                for(i=0; i<n; i++)
                        if (Elem[i]=='1')
                                printf("%d,",i+1);
                printf("\b},");
        }
        printf("\b }\n\n");

        return 0;
}

/* Beispielausgabe:
-----------------------------
| Potenzmenge von {1,...,n} |
-----------------------------

Geben Sie eine Zahl n ein: 6

P={ {},{1},{2},{1,2},{3},{1,3},{2,3},{1,2,3},{4},{1,4},{2,4},{1,2,4},{3,4},{1,3,4},
{2,3,4},{1,2,3,4},{5},{1,5},{2,5},{1,2,5},{3,5},{1,3,5},{2,3,5},{1,2,3,5},{4,5},
{1,4,5},{2,4,5},{1,2,4,5},{3,4,5},{1,3,4,5},{2,3,4,5},{1,2,3,4,5},{6},{1,6},{2,6},
{1,2,6},{3,6},{1,3,6},{2,3,6},{1,2,3,6},{4,6},{1,4,6},{2,4,6},{1,2,4,6},{3,4,6},
{1,3,4,6},{2,3,4,6},{1,2,3,4,6},{5,6},{1,5,6},{2,5,6},{1,2,5,6},{3,5,6},{1,3,5,6},
{2,3,5,6},{1,2,3,5,6},{4,5,6},{1,4,5,6},{2,4,5,6},{1,2,4,5,6},{3,4,5,6},{1,3,4,5,6},
{2,3,4,5,6},{1,2,3,4,5,6} } */
```

3.) potmenge_geordn.c

```
/*      potmenge.c -- Matthias Jauernig, 26.11.03                              */
/*      Programm gibt die Potenzmenge einer natürlichen Zahl n>=1 aus,         */
/*      also alle 2^n Teilmengen der Menge {1,...,n}                           */

#include <stdio.h>

// BITS ist eine symb. Konstante für die max. Anzahl von n (=Anzahl Bits in Dual)
// hier muss <25 gewählt werden, da sonst für die geordnete Ausgabe das zu benutzende
// Array alle Ressourcengrenzen sprengen würde; bei BITS=25 sind das schon
// 26*26^5=11881376 Elemente für a, also 11881376*int=380204032Bi~45MByte Speicher!!!
#define BITS 25

// ausg_menge() - gibt die momentane Menge anhand der Dualzahl aus
void ausg_menge(long int dual, int n){
        int i;
        printf("{");
        // gib den Index der Dualziffern aus, die in der Zahl "1" sind
        for(i=1; i<=n; i++)
                if((dual>>(n-i))&1)
                        printf("%d,", i);
        printf("\b},");
}

// pow() - gibt die Zahl basis^exp zurück und ist Ersatz für pow() aus der math.h
long int pow(int basis, int exp){
        int pot=1,i;

        for(i=1; i<=exp; i++)
                pot*=basis;

        return pot;
}

// anz_eins() - gibt die Anzahl der Einsen der Dualdarstellung von "zahl" zurück
int anz_eins(int zahl, int n){
        int i, z=0;
        for(i=1; i<=n; i++)
                if((zahl>>(n-i))&1)
                        z++;
        return z;
}

// main() -------------------------------------------------------------------------
int main(void){
        int n, i, j, k;
        // a[i][j] speichert die unterschiedlichen Zahlen j, welche i Bits haben
        // b[i] speichert die Anzahl der Zahlen, die gleich viele Bits i haben
        int a[BITS+1][pow(BITS+1,4)], b[BITS+1];

        printf( "-----------------------------\n"
                "| Potenzmenge von {1,...,n} |\n"
                "-----------------------------\n\n");
        do{
                printf("Geben Sie eine ganze Zahl n>=1, n<%d ein: ", BITS);
                scanf("%d", &n);
        }while((n<1 && printf("Zahl muss größer oder gleich 1 sein!\n\n"))
                || (n>BITS-1 && printf("Zahl muss kleiner %d sein!\n\n", BITS)));

        // erstmal beide Hilfsarrays 0 setzen
        for(j=1; j<=n; j++){
                b[j]=0;
                for(k=1; k<=pow(BITS+1,4); k++)
                        a[j][k]=0;
        }
```

```
        printf("\nP({1,...,%d})={ {},", n);
        // gruppiere alle Komb. so, dass sie Gruppen gleicher Bitanzahl bilden
        for(j=1; j<=pow(2,n)-1; j++)
                a[anz_eins(j,n)][++b[anz_eins(j,n)]]=j;

        // äußere Schleife: gib Teilmengen gleicher Kardinalität gemeinsam aus
        // innere Schleife: gib jede Teilmenge in der Gruppe gleicher Kardinalität
        // (=gleicher Bitanzahl) einzeln aus, fange mit der kleinsten an...
        for(i=1; i<=n && a[i][1]; i++)
                for(j=1; b[i]-j+1>=1; j++)
                        ausg_menge(a[i][b[i]-j+1], n);
        printf("\b }\n\n");

        return 0;
}
/* Beispielausgabe
------------------------------
| Potenzmenge von {1,...,n} |
------------------------------

Geben Sie eine ganze Zahl n>=1, n<25 ein: 6

P({1,...,6})={{},{1},{2},{3},{4},{5},{6},{1,2},{1,3},{1,4},{1,5},{1,6},{2,3},{2,4},
{2,5},{2,6},{3,4},{3,5},{3,6},{4,5},{4,6},{5,6},{1,2,3},{1,2,4},{1,2,5},{1,2,6},
{1,3,4},{1,3,5},{1,3,6},{1,4,5},{1,4,6},{1,5,6},{2,3,4},{2,3,5},{2,3,6},{2,4,5},
{2,4,6},{2,5,6},{3,4,5},{3,4,6},{3,5,6},{4,5,6},{1,2,3,4},{1,2,3,5},{1,2,3,6},
{1,2,4,5},{1,2,4,6},{1,2,5,6},{1,3,4,5},{1,3,4,6},{1,3,5,6},{1,4,5,6},{2,3,4,5},
{2,3,4,6},{2,3,5,6},{2,4,5,6},{3,4,5,6},{1,2,3,4,5},{1,2,3,4,6},{1,2,3,5,6},
{1,2,4,5,6},{1,3,4,5,6},{2,3,4,5,6},{1,2,3,4,5,6} } */
```

Zu 24.) *Monte-Carlo-Methode* zur näherungsweisen Berechnung von π

C-Programm, welches die Monte-Carlo-Methode simuliert

```c
/*      monte_carlo.c -- Matthias Jauernig, 28.11.03                        */
/*      Programm berechnet einen näherungsweisen Wert für Pi anhand der     */
/*      "Monte-Carlo-Methode"                                               */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

long double quad(double wert){
        return wert*wert;
}

int main(void){
        unsigned long int i, n, inkreis=0;
        double x, y;
        srand(time(NULL));

        printf( "\nMonte-Carlo-Methode zur naeherungsweisen Bestimmung von Pi\n"
                "-----------------------------------------------------------\n\n");
        do {
                printf("Anzahl der Durchlaeufe (>=1000): ");
                scanf("%lu", &n);
        }while(n<1000 && printf("Zahl muss groesser gleich 1000 sein!\n\n"));

        for(i=1; i<=n; i++){
                x=(double)rand()/RAND_MAX;
                y=(double)rand()/RAND_MAX;
                if(quad(x-1)+quad(y-1) <= 1.0)
                        inkreis++;
        }
        printf("Naeherungsweiser Wert fuer Pi: %10.8F\n\n", 4.0*inkreis/n);

        return 0;
}
```

→ **Testausgaben des Programms:**

```
Anzahl der Durchlaeufe (>=1000): 1000
Naeherungsweiser Wert fuer Pi: 3.196000

Anzahl der Durchlaeufe (>=1000): 10000
Naeherungsweiser Wert fuer Pi: 3.174400

Anzahl der Durchlaeufe (>=1000): 100000
Naeherungsweiser Wert fuer Pi: 3.138600

Anzahl der Durchlaeufe (>=1000): 1000000
Naeherungsweiser Wert fuer Pi: 3.143628

Anzahl der Durchlaeufe (>=1000): 10000000
Naeherungsweiser Wert fuer Pi: 3.142010

Anzahl der Durchlaeufe (>=1000): 100000000
Naeherungsweiser Wert fuer Pi: 3.141574
```