**Aufgabe 37**

a)

#include <stdio.h>

#include <stdlib.h>

int \*cpy\_array(int v[], int size);

int main(void)

{

int i;

int v[] = {7, 8, 9, 10};

int size = sizeof(v) / sizeof(int);

int \*pointer = cpy\_array(v, size);

if (pointer == NULL) {

printf("Speicherreservierung fehlgeschlagen\n");

return 0;

}

for(i = 0; i < size; i++) {

printf("w[%d] = %d\n", i, \*(pointer + i));

}

free(pointer);

pointer = NULL;

return 0;

}

int \*cpy\_array(int v[], int size)

{

int i;

int \* ptr = (int\*)malloc(size \* sizeof(int));

if(ptr == NULL) {

return ptr;

}

for(i = 0; i < size; i++) {

\*(ptr + i) = v[i];

}

return ptr;

}

b)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char \*safecat(char \*s, const char \*ct);

int main(void)

{

int i;

char kette1[] = "Pastewka";

char kette2[] = "Stromberg";

int laenge = strlen(kette1) + strlen(kette2);

char\* pointer = safecat(kette1, kette2);

if(pointer == 0){

printf("Speicher konnte nicht erzeugt werden\n");

return 0;

}

for(i = 0; i < laenge; i++){

printf("%c", \*(pointer + i));

}

printf("\n");

free(pointer);

pointer = NULL;

return 0;

}

char \*safecat(char \*s, const char \*ct)

{

int i;

int laenge = strlen(s) + 1 + strlen(ct);

int laenge1 = strlen(s) + 1;

char \*ptr = (char\*)malloc(laenge \* sizeof(char));

if (ptr == 0){

return ptr;

}

for(i = 0; i < laenge1; i++){

\*(ptr + i) = s[i];

}

strcat(ptr, ct);

return ptr;

}

**Aufgabe 38**

e)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char \*orakel(void);

int main(void)

{

char \*s;

s = orakel();

if (s == NULL) {

printf("main: Speicherfehler in orakel");

return 1;

}

printf("%s", s);

free(s);

return 0;

}

char \*orakel(void)

{

char \*s, \*t;

s = (char\*) malloc(15 \* sizeof(char)); /\*Da "Six Feet Under" 14 Zeichen + \0 sind muss man 15 chars reservieren\*/

if (s == NULL){

return NULL;

}

t = (char\*) malloc(7 \* sizeof(char));

if (t == NULL) {

return NULL;

}

strcpy(s, "Six Feet");

strcpy(t, " Under");

strcat(s, t);

free(t);

return s;

}

**Aufgabe 39**

a)

#include <stdlib.h>

#include <string.h>

int safecpy(char \*\*t, char \*s)

{

if(t == NULL){

return 0;

}

\*t = (char\*)malloc((strlen(s) + 1) \* sizeof(char));

if (\*t == NULL){

return 0;

}

strcpy(\*t,s);

return 1;

}

b)

#include <stdio.h>

#include <stdlib.h>

int main(void)

{

int i;

char \*pointer;

char string[11];

char c;

double ret;

for(i = 0; i < 10; i++){

c = getchar();

if (c == '\n'){

string[i] = '\0';

break;

}

string[i] = c;

}

ret = strtod(string, &pointer);

printf("%f\n", ret);

printf("%s\n", pointer);

return 0;

}

c)

#include <math.h>

double my\_frexp(double x, int \*exponent)

{

int i;

int count;

for(i = 0; fabs(x) >= 1; i++) {

x = x / 2;

count ++;

\*exponent = count;

}

return x;

}

**Aufgabe 40**

a)

#include <stdlib.h>

int \*\*matrix\_cpy( int \*\*m, int ze, int sp)

{

int i;

int j;

int k;

int \*\*pointer;

pointer =(int\*\*) malloc(ze \* sizeof(int \*));

if (m == NULL) {

return NULL;

}

for(j = 0; j < sp; j++){

pointer[j] = (int\*)malloc(sizeof(int));

if (pointer[j] == NULL){

for(k = 0; k < j; k++){

free(pointer[k]);

}

free(pointer);

return NULL;

}

}

for(i = 0; i < ze; i++){

pointer[i]= m[i];

}

return m;

}

b)

#include <stdlib.h>

#include <string.h>

#define N 3

void matrix\_mult\_vek(int a[][N], int x[], int b[], int n, int m)

{

int i;

int j;

int y[N] = {0, 0};

for (i = 0; i < n; i++){

for(j = 0; j < m; j++){

y[i] += (a[i][j] \* x[j]);

}

b[i] = y[i];

}

}

c)

void stringlist\_rev(char \*list[], int n)

{

char \*helplist[1];

int i;

for(i = 0; i < n / 2; i++){

helplist[0] = list[i];

list[i] = list[n - i - 1];

list[n - i -1] = helplist[0];

}

}

d)

void matrix\_print(int \*m, int ze, int sp)

{

int i;

int j;

for(i = 0; i < ze; i++){

for(j = 0; j < sp; j++){

printf("%8.d", \*(m++));

printf(" ");

}

printf("\n");

}

}

e)

# include <stdio.h>

int main(void)

{

int \*v = NULL;

int n = 7;

int res;

int \*x = v;

int \*y = v + n;

if(array\_create(&v, n) == '0'){

printf("Fehler");

return 1;

}

if(array\_create(&v + n, n) == '0'){

array\_destroy(v);

printf("Fehler");

return 2;

}

array\_init(v, n);

array\_init((v + n), n);

res = mult\_array(x, y, n);

array\_destroy(v);

array\_destroy(v + n);

printf("Das Ergebnis ist: %d\n", res);

return 0;

}