**DYN. SPEICHERPLATZ ANFORDERN & FELD VERDOPPELN**

#include <stdio.h>

#include <stdlib.h>

//Dynamisch Speicherplatz anfordern und Feld verdoppeln

int\* twiceret(int arr[], int length){

//dynamisch Speicher anfordern

int \*twice = (int \*) malloc(length \* sizeof(int));

int i;

//Feldelemente kopieren und verdoppeln.

for(i = 0; i<length; i++)

twice[i] = 2\* arr[i];

return twice;

}

int main(void){

int arr[] = {1,2,3};

const int length = 3;

//Verdoppeltes arr wird in doublearr gespeichert

int \*doublearr = twiceret(arr, length);

int i; //Ausgabe des doppelten arr.

for(i=0; i<length; i++)

printf("[%i] ",doublearr[i]);

return 0;

}

**FELDER ADDIEREN**

//Dynamisch Speicher anfordern und Felder addieren

int\* vadd(int arr1[], int arr2[], int length){

int \*addarr = (int \*) malloc(length \* sizeof(int));

int i;

//arr1 und arr2 int addarr addieren:

for(i=0; i<length; i++)

addarr[i] = arr1[i] + arr2[i];

return addarr;

}

**2DIM FELD DYN. ERZEUGEN & NxN Optik erstellen**

//2DIM feld dynamisch erzeugen und nxn Optik erstellen

int\*\* keysum(int n){

//Speicher für 2DIM Feld anfordern

int \*\*keysumarr = (int \*\*) malloc(n \* sizeof(int));

int i, j;

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

keysumarr[i] = (int \*) malloc(n\*sizeof(int));

for(j=0; j<n; j++)

keysumarr[i][j] = i+j;

}

return keysumarr;

}

**3DIM FELD DYN. ERZEUGEN**

//3DIM Feld --Speicher anfordern und ausgeben

int\*\*\* xyzkeysum(int x, int y, int z){

int \*\*\*xyzarr = (int \*\*) malloc(z \* sizeof(int));

int i,j,k;

//z ist anzahl der 2DIM Felder....x+y spannen 2D Feld auf

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

xyzarr[i] = (int \*\*) malloc(x\*sizeof(int));

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

xyzarr[i][j] = (int \*) malloc(y\*sizeof(int));

for(k=0; k<y; k++)

xyzarr[i][j][k] = i+j+k;

}

}

return xyzarr;

}

int main(void){

int x,y,z;

int i,j,k;

printf("Bitte geben Sie x ein: ");

scanf("%i", &x);

printf("Bitte geben Sie y ein: ");

scanf("%i", &y);

printf("Bitte geben Sie z ein: ");

scanf("%i", &z);

int \*\*\*arr = xyzkeysum(x,y,z);

//Ausgabe des 3DIM Felds:

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

printf("\n\n");

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

printf("\n");

for(k=0; k<y; k++)

printf("[%i] ", arr[i][j][k]);

}

}

return 0;

}

**VIGENERE QUADRAT DYNAMISCH**

//Erstelle das VigenereQuadrat als 2DIM Feld

//VigenereQuadrat: Horizontal ist Nachricht und vertikal Codewort

char\*\* vigenereQuadrat(){

char abc[26] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

char \*\*vigenerearr = (char \*\*) malloc(26\*sizeof(char\*));

//bei sizeof muss nach char\* ein "\*", da er 26 mal ein 1Dim Feld im 2Dim Feld bereitstellen soll

int i,j,l;

for(i=0; i<26; i++){

vigenerearr[i] = (char \*) malloc(26\*sizeof(char));

j=0;

l=0;

do{

if(j+i <=25)

vigenerearr[i][j] = abc[j+i];

if(j+i >25){

vigenerearr[i][j] = abc[l];

l++; //Um wieder bei A anzufangen wenn einmal durch

}

j++;

}while(j<26);

}

return vigenerearr;

}

**PASCAL DREIECK**

int\*\* pascal(int n){

int \*\*arr = (int \*\*) malloc((n+1) \* sizeof(int\*));

int row, column;

int i;

//Es soll ein dynamisches 2DIM Feld als Pascaldreieck erzeugt werden.

//Die Anzahl der Feldelemente ist unterschiedlich-->Deshalb wird immer

//an der ersten Feldstelle die Information über die Länge gespeichert.

for(row=0; row<n+1; row++){

arr[row] = (int \*) malloc((row+2) \* sizeof(int)); //+2, da Feldlänge und +1 noch dazu kommt

arr[row][0] = row+2; //Länge speichern..

arr[0][1] = 1;

if(row==1){

arr[1][1] = 1;

arr[1][2] = 1;

}

for(column=1; column<row+2; column++){

if(row<2)

break;

if(column==1 || column==row+1)

arr[row][column] = 1;

else{

arr[row][column] = arr[row-1][column-1] + arr[row-1][column];

}

}

}

return arr;

}

**MINIRECHNER SCANF**

//Minirechner durch sscanf

int main(void)

{

char string[100];

int operand1, operand2;

char operation;

printf("Operand1 Operation Operand2\n");

do{

printf("Ausdruck: ");

gets(string);

sscanf(string,"%i %c %i", &operand1, &operation, &operand2);

if(strcmp(string, "quit")){

printf("%s = ", string);

switch(operation){

case '+': printf("%i", (operand1 + operand2));

break;

case '-': printf("%i", (operand1 - operand2));

break;

case '\*': printf("%i", (operand1 \* operand2));

break;

case '/': printf("%i", (operand1 / operand2));

break;

}

}

printf("\n\n");

}while(strcmp(string, "quit"));

printf("Ade\n");

return 0;

}

**TEXTDATEI KOPIEREN UND AUSGEBEN**

#include <stdio.h>

int main(void)

{

FILE \*file;

char str[100];

if((file = fopen("text.txt", "r")) == NULL){//Datei öffnen zum lesen

printf("File does not exist!\n");

return -1;

}

else{

FILE \*file2; //2 Datei für die Kopie

if((file2 = fopen("copy.txt", "w")) == NULL){ //Datei erzeugen zum schreiben

printf("Cannot create file!\n");

}

else{

while(!feof(file)){ //feof: File end of file --geht bis zum ende der text.txt durch

fgets(str, 100, file); //Holt sich 100 Zeichen aus text.txt

fprintf(file2, str); //Schreibt diese 100 Zeichen in die copy.txt

printf("%s", str);

}

}

fclose(file);

fclose(file2);

return 0;

}

}

**AUS DATEI LESEN & VERGLEICHEN**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

//Das Programm liest aus einer Textdatei Namen und die Anzahl der Namen ein und vergleicht diese mit

//eingegebenen Namen. Je nach Geschlecht werden dann die Anreden kommen.

int main(void)

{

FILE \*male;

FILE \*female;

int tmp1, tmp2, length, counter=0;

char str[20];

char \*\*maleArr;

char \*\*femaleArr;

male = fopen("male.txt", "r");

female = fopen("female.txt", "r");

if(male == NULL)

printf("File 'male.txt' does not exist!\n");

if(female == NULL)

printf("File 'female.txt' does not exist!\n");

if((male != NULL) && (female != NULL)){

while(!feof(male)){//Geht nacheinander bis zum Ende der Datei jede Zeile durch-> fscanf

if(counter == 0){

fscanf(male, "%i", &tmp1); //Zuerst Anzahl der Namen in .txt Datei holen

maleArr = (char \*\*) malloc(tmp1 \* sizeof(char\*)); //daraus dyn. Feld erstellen --2DIM

counter++;

}

else{

fscanf(male, "%s\n", str); //Einen Namen aus der Datei holen

length = strlen(str); //Name ist nun in str--> Länge des Namen ermitteln

maleArr[counter-1] = (char \*) malloc(length \* sizeof(char)); //Mit Länge dyn. Feld erzeugen.

strcpy(maleArr[counter -1], str);

counter++;

}

}

counter=0;

while(!feof(female)){

if(counter == 0){

fscanf(female, "%i", &tmp2); //tmp2 hat Anzahl der Namen, da am Anfang in .txt

femaleArr = (char \*\*) malloc(tmp2 \* sizeof(char\*));

counter++;

}

else{

fscanf(female, "%s\n", str);

length = strlen(str);

femaleArr[counter-1] = (char \*) malloc(length \* sizeof(char));

strcpy(femaleArr[counter -1], str);

counter++;

}

}

}

int i, p1, p2;

printf("Gelesene weibliche Vornamen: ");

for(i=0; i<tmp2; i++){

printf("%s, ", femaleArr[i]);

}

printf("\nGelesene männliche Vornamen: ");

for(i=0; i<tmp1; i++){

printf("%s, ", maleArr[i]);

}

printf("\nGeben Sie einen Namen ein: ");

gets(str);

//Geht alle Namen durch und vergleicht mit eingeg. Namen

//Liefert Zeiger wenn enthalten, sonst Nullzeiger!!

for(i = 0; i < tmp1; i++){

p1 = strstr(str, maleArr[i]);

if(p1 !=0)

break;

}

for(i = 0; i < tmp2; i++){

p2 = strstr(str, femaleArr[i]);

if(p2 !=0)

break;

}

if(p2 !=0)

printf("%s %s\n", "Ms.", str);

if(p1 !=0)

printf("%s %s\n", "Mr.", str);

if(p1 == 0 && p2 == 0)

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

return 0;

}

**ARR IN .dat DATEI SCHREIBEN & LESEN**

//Schreiben und lesen mit dynamischen Feld!

//arr wird dynamisch erzeugt und initialisiert--> in erzeugte Textdatei geschrieben

void writebin(void){

FILE \*file;

int i;

int \*\*arr = (int \*\*) malloc(3\*sizeof(int\*));

for (i=0; i<3; i++)

arr[i] = (int \*) malloc(2\*sizeof(int));

arr[0][0] = 1; arr[0][1] = 2; arr[1][0] = 3;

arr[1][1] = 4; arr[2][0] = 5; arr[2][1] = 6;

if((file = fopen("bin.dat", "wb")) == NULL){

printf("Cannot create file!\n");

}

else{

for(i=0; i<3; i++) //Zeilenweise schreiben

fwrite(arr[i], sizeof(int), 2, file);

}

fclose(file);

}

//Feld wird dynamisch erzeugt und aus Datei in arr gelesen

void readbin(){

FILE \*file;

int i;

int \*\*arr = (int \*\*) malloc(3\*sizeof(int\*));

for (i=0; i<3; i++)

arr[i] = (int \*) malloc(2\*sizeof(int));

if((file = fopen("bin.dat", "rb")) == NULL){

printf("Cannot open file!\n");

}

else {

for (i=0; i<3; i++) // Zeilenweise lesen

fread(arr[i], sizeof(int), 2, file);

}

fclose(file);

for(i=0; i<3; i++){

printf("[%i] [%i]\n", arr[i][0], arr[i][1]);

}

}

**FUNKTIONEN DATEI LESEN & SCHREIBEN & …**

//2DIM feld dynamisch erzeugen und nxn Optik erstellen

int\*\* keysum(int n){

//Speicher für 2DIM Feld anfordern

int \*\*keysumarr = (int \*\*) malloc(n \* sizeof(int));

int i, j;

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

keysumarr[i] = (int \*) malloc(n\*sizeof(int));

for(j=0; j<n; j++)

keysumarr[i][j] = i+j;

}

return keysumarr;

}

//Funktion speichert ein nxn Feld in einer bin.dat File

void safeInFile(int \*\*arr, int n){

FILE \*file;

int i;

if((file = fopen("bin.dat", "wb")) == NULL){

printf("Cannot create file!\n");

}

else{

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

fwrite(arr[i], sizeof(int), n, file);

fclose(file);

}

}

//Funktion überschreibt das Feld mit 0

void overwriteFile(int \*\*arr, int n){

int i,j;

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

for(j=0; j<n; j++)

arr[i][j] = 0;

}

}

//Funktion lest arr aus einer Datei und gibt es aus

void readFile(int \*\*arr, int n){

FILE \*file;

int i;

if((file = fopen("bin.dat", "rb")) == NULL)

printf("Cannot open file!\n");

else{

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

fread(arr[i], sizeof(int), n, file);

}

fclose(file);

}

}

//Funktion gibt \*\*arr aus

void print(int \*\*arr, int n){

int i, j;

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

printf("\n");

for(j=0; j<n; j++)

printf("[%i] ", arr[i][j]);

}

}

**STRUCT**

struct Vec3f{

float x, y, z;

void init(float xval, float yval, float zval){

x = xval;

y = yval;

z = zval;

}

void add(Vec3f v){

x += v.x;

y += v.y;

z += v.z;

}

void sub(Vec3f v){

x -= v.x;

y -= v.y;

z -= v.z;

}

float scalarProduct(Vec3f v){

float scalpro;

scalpro = x \* v.x + y \* v.y + z \* v.z;

return scalpro;

}

void crossProduct(Vec3f v){

int a=x, b=y, c=z;//Werte retten da sonst während rechnen überschrieben wird!!

x = b\*v.z - c\*v.y;

y = c\*v.x - a\*v.z;

z = a\*v.y - b\*v.x;

}

char\* toString(){

char \*str = new char[100];

sprintf(str, "x=%f\ny=%f\nz=%f\n",x,y,z);

printf("%s",str);

}

};

int main()

{

Vec3f v1, v2;

v1.init(3,-1,2);

v2.init(5,9,-2);

printf("v1 + v2 =\n");

v1.add(v2);

v1.toString();

}

**LISTE MIT PUNKTEN**

#include <stdio.h>

typedef struct \_Point {

float x;

float y;

struct \_Point \*next;

} Point;

void printList(Point \*headOfList) {

Point \*currentElem = headOfList;

while (currentElem != NULL) {

printf("x: %.2f, y: %.2f\n",currentElem->x, currentElem->y);

currentElem = currentElem->next;

}

printf("EOL\n");

}

void removeLastElement (Point \*\*headOfList) {

Point \*currentElem = \*headOfList;

Point \*previousElem = currentElem;

if (\*headOfList != NULL) {

while (currentElem->next != NULL) {

previousElem = currentElem;

currentElem = currentElem->next;

}

free(currentElem);

if (previousElem != currentElem){

previousElem->next = NULL;

}

else { // Verweis auf Listenkopf löschen

\*headOfList = NULL;

}

}

}

void append(Point \*\*headOfList, float x, float y) {

Point \*currentElem = \*headOfList;

// Listenelement allozieren

Point \*newPoint = (Point \*) malloc (sizeof(Point));

newPoint->x = x; // Listenelement Daten zuweisen

newPoint->y = y;

newPoint->next = NULL; // Nullzeiger: Markiert Ende der Liste

// Falls Liste leer: Listenkopf erzeugen

if (currentElem == NULL) {

\*headOfList = newPoint;

}

else { // Anderenfalls neues Element anfügen

while (currentElem->next != NULL) {

currentElem = currentElem->next;

}

currentElem->next = newPoint;

}

}

**MAIN:** Point \*headOfList = NULL;

**LISTE MIT PERSONEN**

#include <stdio.h>

typedef struct \_Person {

char \*name;

int age;

float size;

struct \_Person \*next;

} Person;

void printpList(Person \*headOfList) {

Person \*currentElem = headOfList;

while (currentElem != NULL) {

printf("Name: %s\nAlter: %i\nGroesse: %.2f\n\n",currentElem->name, currentElem->age, currentElem->size);

currentElem = currentElem->next;

}

printf("EOL\n");

}

char\* newName(const char \*name){

char \*newName = (char \*)malloc((strlen(name)+1)\*sizeof(char));

sprintf(newName, "%s", name);

return newName;

}

void removeLastElement (Person \*\*headOfList) {

Person \*currentElem = \*headOfList;

Person \*previousElem = currentElem;

if (\*headOfList != NULL) {

while (currentElem->next != NULL) {

previousElem = currentElem;

currentElem = currentElem->next;

}

free(currentElem);

if (previousElem != currentElem){

previousElem->next = NULL;

}

else { // Verweis auf Listenkopf löschen

\*headOfList = NULL;

}

}

}

void insert(Person \*\*headOfList, char\* name, int age, float size){

Person \*currentElem = \*headOfList;

// Listenelement allozieren

Person \*newPerson = (Person \*) malloc (sizeof(Person));

newPerson->name = name; // Listenelement Daten zuweisen

newPerson->age = age;

newPerson->size = size;

newPerson->next = NULL; // Nullzeiger: Markiert Ende der Liste

// Falls Liste leer: Listenkopf erzeugen

if (currentElem == NULL) {

\*headOfList = newPerson;

}

else { // Anderenfalls neues Element anfügen

while (currentElem->next != NULL) {

currentElem = currentElem->next;

}

currentElem->next = newPerson;

}

} **MAIN:** Person \*headOfList = NULL; insert(&headOfList, newName("Anton"), 21, 1.82);

**IN LISTE GEZIELT SUCHEN ODER GEZIELT LÖSCHEN**

//Die Funktion bekommt den Name oder Alter oder size und sucht danach. Es gibt den Zeiger auf das struct zurück.

Person \*pfind(Person \*\*headOfList, char\* name, int age, float size){//Gibt Zeiger auf gefundenes Listenelement zurück

Person \*currentElem = \*headOfList;

Person \*previousElem = currentElem;

if(name != NULL){//Name suchen

while(currentElem!=NULL && (0!=strcmp(currentElem->name, name))){//strcmp liefert 0 wenn gleich

previousElem = currentElem;

currentElem = currentElem->next;

}

return currentElem;

}

if(name==NULL && age>=0){//Alter suchen

while(currentElem!=NULL && (currentElem->age != age)){

previousElem = currentElem;

currentElem = currentElem->next;

}

return currentElem;

}

if(name==NULL && age<0 && size>0){//Groesse suchen

while(currentElem!=NULL && (currentElem->size != size)){

previousElem = currentElem;

currentElem = currentElem->next;

}

return currentElem;

}

}

//Die Funktion bekommt den Name oder Alter oder size und löscht gefundenes Element. Gibt 0 oder 1 zurück.

int pdelete(Person \*\*headOfList, char\* name, int age, float size){

Person \*currentElem = \*headOfList;

Person \*previousElem = currentElem;

//Sucht und wenn gefunden wird aktuelles Element in currentElem und das davor in previousElem gespeichert

if(name != NULL){//Name suchen

while(currentElem!=NULL && (0!=strcmp(currentElem->name, name))){//strcmp liefert 0 wenn gleich

previousElem = currentElem;

currentElem = currentElem->next;

}

}

if(name==NULL && age>=0){//Alter suchen

while(currentElem!=NULL && (currentElem->age != age)){

previousElem = currentElem;

currentElem = currentElem->next;

}

}

if(name==NULL && age<0 && size>0){//Groesse suchen

while(currentElem!=NULL && (currentElem->size != size)){

previousElem = currentElem;

currentElem = currentElem->next;

}

}

if(currentElem != NULL){

if(previousElem == currentElem){//wenn bei Listenkopf stehen geblieben

\*headOfList = currentElem->next;

free(currentElem);

}

else{//Wenn am Listenende oder zwischendrin!

previousElem->next = currentElem->next;

free(currentElem);

}

return 1; //Wenn loeschen erfolgreich!

}

else

return 0;

}

**STAPEL MIT STRUCT**

typedef struct ListNode{

int value;

struct ListNode \*next;

}List1;

struct Stack{

List1 \*list = NULL;

void push(int val){//Mit push kann die Integerzahl in val auf den Stapel gelegt werden.

List1 \*newValue= new List1;

newValue->value = val;

newValue->next = NULL;

if(list == NULL){

list = newValue;

}

else{

newValue->next = list;

list = newValue;

}

}

int pop(int \*val){

List1 \*currentElem = list;

if(list == NULL){

return 0;

}

else{

\*val = list->value;

if(list != NULL){

list = list->next;

delete currentElem;

return 1;

}

}

}

int empty(){

if(list == NULL){

return 1;

}

else{

return 0;

}

}

};

int main()

{

Stack stapel1;

……..

return 0;

}

**RINGBUFFER LIST**

typedef struct \_List{

char letter;

struct \_List \*next;

}List;

struct Ringbuffer{

List \*InPointer = NULL; //Zeiger auf Listenanfang

List \*OutPointer = NULL; //Zeiger auf Listenende

List \*write;

List \*read;

void init(int length){//Erstellt dyn. Puffer der Länge length

int i;

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

List \*newChar = new List;

if(InPointer == NULL){//wenn noch kein Element in Liste

InPointer = newChar;//Wandert nach rechts

OutPointer = newChar;//Immer am linken Ende

}

else{ //Wenn schon Elemente in Liste.

InPointer->next = newChar;

InPointer = newChar;

}

}

write = OutPointer;//damit man Anfang der Liste weiß

read = OutPointer;

}

void push(char val){//WRITE: Schreibt zeichen an Position Ende (von links nach rechts)->Wenn Pufferende, dann wieder zu Ende (nach links)

if(write != InPointer){//Solange nicht ganz rechts

write->letter = val;

write = write->next; //Zum nächsten Element springen

}

else{

write->letter = val;

write = OutPointer; //wenn am Anfang, dann wieder zurück zu Ende-> OUtPointer (nach links)

}

}

char pop(){//READ: Lesen und zurückgeben eines Zeichens ->read läuft Liste durch und gibt aus, bis zu InPointer

char c;

if(read != InPointer){

c = read->letter;

read = read->next;

}

else{

c = read->letter;

read = OutPointer;

}

return c;

}

};

**UEBERLADEN**

//Funktionen überladen--> Sucht sich je nach float oder int Funktion aus und tauscht

//Variante1:

void flip(int &num1, int &num2){

int tmp;

if(num2>num1){

tmp = num1;

num1 = num2;

num2 =tmp;

}

}

void flip(float &num1, float &num2){

float tmp;

if(num2>num1){

tmp = num1;

num1 = num2;

num2 =tmp;

}

}

//Bessere Lösung mit Template!

template<typename TYPE>

void flip(TYPE &num1, TYPE &num2){

TYPE tmp;

if(num2>num1){

tmp = num1;

num1 = num2;

num2 =tmp;

}

}

int main()

{

float a = 5.5;

float b = 9.9;

flip(a,b);

cout << "a: " << a << "\nb: " << b << endl;

return 0;

}

**TEMPLATE**

//Template das dynamisches Feld erzeugt und Elemente umgekehrt einfügt:

//Gibt dann ein Zeiger auf das neue Feld zurück..

template<typename TYPE1>

TYPE1 \*change(TYPE1 z[], int length){

TYPE1 \*helparr = new TYPE1[length];

int i;

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

helparr[i] = z[length-1-i];

}

return helparr;

}

int main()

{

const int length = 6;

//Hier können beide int z.B. in ein float geändert werden.

//Die Funktion reagiert darauf automatisch!!

int z[] = {1,2,3,4,5,6};

int \*newarr= change(z,length);

cout << "newarr[]: \n"<< endl;

int i;

for(i=0; i<length; i++)

cout << newarr[i]<<endl;

return 0;

}

**TEMPLATE CLASS**

#ifndef NUMBERCONTAINER\_HPP

#define NUMBERCONTAINER\_HPP

template <class T>

class NumberContainer {

private:

int length; //Für die Anzahl der im Containerproj gespeicherten Zahlenwerte

T \*carr = new T[length]; //dyn. Feld in dem diese Zahlen gespeichert sind

public:

NumberContainer(int length); //length wieder Anzahl Zahlenwerte

bool set(T val, int pos);

bool get(T &val, int pos);

void flip();//Dreht Zahlenwerte um

};

#include "NumberContainer.tpp"

#endif // NUMBERCONTAINER\_HPP

**TPP:**

template<class T>

NumberContainer<T>::NumberContainer(int length) {

this->length = length;

}

template <class T>

bool NumberContainer<T>::set(T val, int pos) {

if(pos >= 0 && pos < length){

carr[pos] = val;

return true;

}

else{

return false;

}

}

**…..usw**

int main()

{

int tmpval, i,j,k;

bool state;

cout << "Wie viele Feldelemente: ";

cin >> i;

NumberContainer<int> first(i); //i elemente vom Typ int..

……..

**KOMMENTARE AUS DATEI EXTRAHIEREN**

// reading a text file

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main () {

string line;

ifstream myfile ("example.cpp");

if (myfile.is\_open())

{

while ( getline (myfile,line) )

{

if(line.find("//", 0) <= line.length()) //Sucht ob // innerhalb der line vorhanden ist

cout << line.substr(line.find("//",0)) << endl;

}

myfile.close();

}

else cout << "Unable to open file" << endl;

return 0;

}

**TEMPLATE STACK MIT KOPIERKINSTRUKTOR+ EQUALS**

#ifndef STACK\_HPP

#define STACK\_HPP

template <class T>

class Stack{

public:

bool push(T val);

bool pop(T &val);

void clear();

bool empty();

bool full();

bool equals(const Stack &s);

Stack(int psize);

~Stack();

Stack(Stack & object);

private:

bool state;

int index;

T \*arr;

int size;

};

#include "Stack.tpp"

#endif // STACK\_HPP

#include <iostream>

using namespace std;

template <class T>

Stack<T>::Stack(int psize) { //Konstruktor

this->size = psize;

arr = new T[size];

index = 0;

}

template <class T>

Stack<T>::~Stack() { //Destruktor

cout << "Destruktor aufgerufen. arr wird geloescht" << endl;

delete [] arr;

}

template <class T>

Stack<T>::Stack(Stack & object) {

arr = new T[object.size];

index = 0;

this->size = object.size;

int i;

T tmpval;

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

object.pop(tmpval); //werte von dem übergebenen Objekt holen

arr[object.size-1-i] = tmpval; //Werte von hinternnach vorne in Feld schreiben

index++; //damit im kopierten Objekt auch wieder der Index mit Anzahl Feldelem steigt

}

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

object.push(arr[i]); //Werte nun wieder in zu kopierendes Feld speichern da bei pop geloescht wird

}

}

template <class T>

bool Stack<T>::equals(const Stack &s) { //Mit Referenz auf zu vergleichendes Objekt

int c;

if(size == s.size) {

if(index == s.index){

for(c=0; c<index+1; c++) {

if(arr[c] != s.arr[c])

return false;

}

return true; // wenn er alles durchlaufen ist muss er true zurück geben

}else

return false;

}else

return false;

}

template <class T>

bool Stack<T>::push(T val){

state = full(); //Wert intern speichern bevor rückgabe!

if(state == false){

arr[index] = val;

index++;

}

return state;

}

template <class T>

bool Stack<T>::pop(T &val){//Soll wenn Stapel voll ein true geben

state = empty();//wie oben!!

if(state == false){ //Nur ausführen wenn Stapel nicht leer!!

val = arr[index-1]; //-1, da index bei 0 anfängt, size nicht!!

index--;

}

return state;

}

template <class T>

bool Stack<T>::empty(){

if(index==0){

cout << "stack empty" << endl;

return true;

}

else

return false;

}

template <class T>

bool Stack<T>::full(){

if(index == size){

cout << "stack full" << endl;

return true;

}

else

return false;

}

template <class T>

void Stack<T>::clear() {

index = 0;

if(empty())

cout << "Feldelemente geloescht." << endl;

}

**MAIN**

template<class T>

bool operator == (Stack<T> &sl, Stack<T> &sr) {

return sl.equals(sr);

}

int main()

{

bool StackState;

int i;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//3. Einen Operator für direkten Vergleich ==

//Der Operator ist in der .hpp und .tpp realisiert

Stack<int> IntStapelA(3);

Stack<int> IntStapelB(3);

int val1;

int arr1[] = {-7, 1, 12, 2};

for(i = 0; i<4; i++){ //Zuerst in IntStapelA 3 Zahlen schreiben

StackState = IntStapelA.push(arr1[i]);

StackState = IntStapelB.push(arr1[i]);

if(StackState)

break;

cout << arr1[i] << " now on top of the stack." << endl;

}

cout << endl;

if(IntStapelA == IntStapelB)

cout << "gleich" << endl;

else

cout << "verschieden" << endl;

return 0;

}

**KOMPOSITION**

**MAIN:**

#include <iostream>

#include "car.h"

using namespace std;

int main()

{

Car car;

car.left.window.rollup();

car.wheel[0].inflate(72);

car.left.close();

car.right.close();

car.left.window.rolldown();

return 0;

}

**CAR**

#ifndef CAR\_H

#define CAR\_H

#include "engine.h"

#include "wheel.h"

#include "door.h"

class Car {

public:

Car();

Engine engine; //KOMPOSITION

Wheel wheel[4]; //KOMPOSITION

Door left, right; //KOMPOSITION

};

#endif // CAR\_H

#include "car.h"

Car::Car():left("Left door"), right("Right door") {}

**DOOR:**

#ifndef DOOR\_H

#define DOOR\_H

#include "window.h"

#include <iostream>

#include <string>

using namespace std;

class Door {

public:

Window window; //<-- KOMPOSITION

void open();

void close();

Door(string whichDoor);

private:

string whichDoor;

};

#endif // DOOR\_H

#include "door.h"

#include <string>

#include <iostream>

using namespace std;

Door::Door(string whichDoor):whichDoor(whichDoor){}

void Door::open() {

cout << whichDoor << " open" << endl;

}

void Door::close() {

cout << whichDoor << " closed" << endl;

}

**ENGINE**

#ifndef ENGINE\_H

#define ENGINE\_H

class Engine {

public:

void start(); // Motor starten

void stop(); // Motor stoppen

void accelerator(int rpm); // Drehzahl ändern

};

#endif // ENGINE\_H

#include "engine.h"

#include <iostream>

using namespace std;

void Engine::start() {

cout << "Engine started" << endl;

}

void Engine::stop() {

cout << "Engine stopped" << endl;

}

void Engine::accelerator(int rpm) {

cout << rpm << "rpm" << endl;

}

**WHEEL**

#ifndef WHEEL\_H

#define WHEEL\_H

class Wheel {

public:

// Luftdruck anpassen

void inflate(float pressure);

};

#endif // WHEEL\_H

#include "wheel.h"

#include <iostream>

using namespace std;

void Wheel::inflate(float pressure) {

cout << "Wheel pressure: " << pressure << endl;

}

**WINDOW**

#ifndef WINDOW\_H

#define WINDOW\_H

class Window{

public:

void rollup(); // Fenster öffnen

void rolldown();

// Fenster schließen

};

#endif // WINDOW\_H

#include "wheel.h"

#include <iostream>

using namespace std;

void Wheel::inflate(float pressure) {

cout << "Wheel pressure: " << pressure << endl;

}

**VERERBUNG**

**MAIN**

int main()

{

Audio m1("An Artist", "A Song", "mp3");

Book m2("An Author", "A Title", "pdf");

Video m3("A director", "A Movie", "webm");

m1.show();

m2.show();

m3.show();

MusicVideo vid("Steffen", "Tom", "Test", "mp7");

vid.show();

return 0;

}

**MEDIUM**

class Medium {

protected:

string title; // name of the item

string format; // mp3, mov, mpg,...

public:

Medium(string title="no", string format="no");

void show();

};

Medium::Medium(string title, string format):

title(title), format(format) {}

void Medium::show() {

cout << title << ", " << format << endl;

}

**AUDIO**

class Audio : public Medium {

protected:

string artist;

// name of the artist

public:

Audio(string artist="no",string title="no", string format="no");

void show();

};

Audio::Audio(string artist, string title, string format):

Medium(title, format), artist(artist) {}

void Audio::show() {

cout << "Artist: " << artist << ", ";

Medium::show();

}

**VIDEO = AUDIO…..**

**MUSICVIDEO**

class MusicVideo:

public Audio, public Video

{

public:

MusicVideo(string artist="no", /\* Audio \*/

string director="no", /\* Video \*/

/\* Medium \*/

string title="no", string format="no");

void show();

};

//Redundanzreduktion

MusicVideo::MusicVideo(string artist, string director,string title, string format):

Audio(artist, title, format),

Video(director, title, format) {}

void MusicVideo::show() {

cout << "Director: " << director << ", ";

Audio::show();

}

**VERERBUNG**

**MAIN**

int main()

{

Reinigungskraft cleaner;

Fahrroboter \*bot[3];

bot[0] = new Saugroboter("Duster");

bot[1] = new Wischroboter("Wiper");

//bot[2] = new Fensterputzroboter("Climber");

for(int i=0; i<2; i++) {

cleaner.benutzen(bot[i]);

}

return 0;

}

**FAHRROBOTER**

class Fahrroboter

{

public:

Fahrroboter(string pDriver);

void Fahren();

void Laden();

virtual char *what*();

protected:

string Driver;

};

Fahrroboter::Fahrroboter(string pDriver): Driver(pDriver) {}

void Fahrroboter::Fahren() {

cout << Driver << " faehrt" << endl;

}

void Fahrroboter::Laden() {

cout << Driver << " laedt seine Akkus" << endl;

}

char Fahrroboter::*what*() {

return 'f';

}

**REINIGUNGSKRAFT**

//Verteilt Aufgaben an Roboter

class Reinigungskraft

{

public:

Reinigungskraft();

void benutzen(Fahrroboter \*r);

};

Reinigungskraft::Reinigungskraft()

{}

void Reinigungskraft::benutzen(Fahrroboter \*r) {

if(r->*what*() == 'w') { //Wischroboter

cout << "Reinigungskraft: Starte Wischroboter" << endl;

Wischroboter \*w = (Wischroboter\*) r;

w->Wischen();

}

if(r->*what*() == 's') {//Saugroboter

cout << "Reinigungskraft: Starte Saugroboter" << endl;

Saugroboter \*s = (Saugroboter\*) r;

s->Saugen();

}

}

**SAUGROBOTER wie WISCHROBOTER**

class Saugroboter : public Fahrroboter

{

public:

Saugroboter(string pDriver);

void Saugen();

char *what*();

};

Saugroboter::Saugroboter(string pDriver): Fahrroboter(pDriver) {}

void Saugroboter::Saugen() {

cout << Driver << " saugt" << endl;

}

char Saugroboter::*what*() {

return 's';

}

**ODER:**

**KLASSE FAHRROBOTER BEFREUNDEN MIT:** friend class Reinigungskraft;

**DANN VIRTUELLE FUNKTON: virtual void arbeiten(); in Oberklasse Fahrroboter und Unterkl. Wisch- und Saugroboter…**

**REINIGUGSKRAFT MACHT DANN EINFACH:**

void Reinigungskraft::benutzen(Fahrroboter &r) {

cout << "Reinigungskraft: Starte " << r.Driver << endl;

r.*arbeiten*();

}

**IN GUI ZEICHNEN**

**MAIN**

void MainWindow::*paintEvent*(QPaintEvent \*event)

{

// Erzeugen eines neuen QPainter-Objekts, welches als

// Ziel der Grafikoperationen das Programmfenster erhält.

QPainter painter(this);

// Zeichnen einer schwarzen Linie mit Strichstärke 2 von

// (x1,y1) = (50,100) bis (x2,y2) = (150,200).

// Für das Einstellen der Stifteigenschaften des QPainter

// wird ein QPen-Objekt erzeugt.

painter.setPen(QPen(Qt::black, 2));

// painter.drawLine(50,100,150,200);

Shapes \*shapes[3];

shapes[0] = new Triangle(&painter, 50);

shapes[1] = new Square(&painter,50);

shapes[2] = new Pentagon(&painter,50);

shapes[0]->*draw*(200,50);

shapes[1]->*draw*(200,150);

shapes[2]->*draw*(200,250);

}

**OBERKLASSE SHAPES**

class Shapes

{

public:

Shapes(QPainter \*painter, int size=0);

virtual void *draw*(int x, int y) =0;

protected:

QPainter \*object;

int size;

};

Shapes::Shapes(QPainter \*painter, int size):

object(painter), size(size) {}

**UNTERKLASSE TRIANGLE; WIE SQUARE & PENTAGON**

class Triangle: public Shapes

{

public:

Triangle(QPainter \*painter, int size);

void *draw*(int x, int y);

};

Triangle::Triangle(QPainter \*painter, int size):

Shapes(painter, size) {}

void Triangle::*draw*(int x, int y) {

object->drawLine(x-size/2, y+size/2, x, y-size/2);

object->drawLine(x, y-size/2, x+size/2, y+size/2);

object->drawLine(x+size/2, y+size/2 ,x-size/2, y+size/2);

}

**GUI PROGRAMMATISCH**

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

private:

//Ui::MainWindow \*ui;

QWidget widget; //Zentrales widget für gui in dem alles plaziert wird

QCheckBox checkbox;

QRadioButton rb1, rb2, rb3, rb4, rb5;

QLineEdit lineedit;

QVBoxLayout vlayout;

QHBoxLayout hlayout1, hlayout2;

QSignalMapper s;

bool enabled;

public slots:

void buttonHandler(int i);

void checkBoxHandler();

};

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent)//, ui(new Ui::MainWindow)

{

//ui->setupUi(this);

hlayout1.addWidget(&checkbox);

hlayout1.addWidget(&lineedit);

hlayout2.addWidget(&rb1);//Pushbuttons

hlayout2.addWidget(&rb2);

hlayout2.addWidget(&rb3);

hlayout2.addWidget(&rb4);

hlayout2.addWidget(&rb5);

vlayout.addLayout(&hlayout1);

vlayout.addLayout(&hlayout2);

widget.setLayout(&vlayout);

this->setCentralWidget(&widget);

this->setWindowTitle("Halloooo");

QObject::connect(&rb1, SIGNAL(clicked()), &s, SLOT(map()));

QObject::connect(&rb2, SIGNAL(clicked()), &s, SLOT(map()));

QObject::connect(&rb3, SIGNAL(clicked()), &s, SLOT(map()));

QObject::connect(&rb4, SIGNAL(clicked()), &s, SLOT(map()));

QObject::connect(&rb5, SIGNAL(clicked()), &s, SLOT(map()));

//Klickereignis im Signal-Mapper auf die Zahlenwerte 1-5 abbilden

s.setMapping(&rb1, 1);

s.setMapping(&rb2, 2);

s.setMapping(&rb3, 3);

s.setMapping(&rb4, 4);

s.setMapping(&rb5, 5);

//SignalMapper mit behandler verbinden.

QObject::connect(&s, SIGNAL(mapped(int)),this, SLOT(buttonHandler(int)));

QObject::connect(&checkbox, SIGNAL(clicked()), this, SLOT(checkBoxHandler()));

bool enabled = true;

}

void MainWindow::buttonHandler(int i) {

ostringstream strStream;

strStream << "Kanal " << i << " gewaehlt";

lineedit.setText(QString::fromStdString(strStream.str()));

}

void MainWindow::checkBoxHandler() {

if(enabled) {

this->lineedit.setEnabled(false);

this->rb1.setEnabled(false);

this->rb2.setEnabled(false);

this->rb3.setEnabled(false);

this->rb4.setEnabled(false);

this->rb5.setEnabled(false);

enabled = false;

}

else {

this->lineedit.setEnabled(true);

this->rb1.setEnabled(true);

this->rb2.setEnabled(true);

this->rb3.setEnabled(true);

this->rb4.setEnabled(true);

this->rb5.setEnabled(true);

enabled = true;

}

}

MainWindow::~*MainWindow*()

{

//delete ui;

}