

Informatik
Klausur
Sommersemester 2019

Software Projekt

Juli 2019

Klausurzeit: 90min

HINWEISE

1. Die Klausur besteht aus 4 Aufgaben auf 12 Seiten
2. Hilfsmittel sind nicht zugelassen
3. Die Lösungen sind in den Klausurbogen einzutragen, gegebenenfalls sind die Rückseiten der Blätter mit zu benutzen.
4. Eigene Zusatzblätter sind nicht erlaubt
5. Schreiben Sie bitte auf alle Blätter ihren Namen und Matrikelnummer

Name: _____

Matrikelnummer: _____

Aufgabe	Maximale Punktzahl	Erreichte Punkte
A1	10	
A2	15	
A3	45	
A4	20	
Summe	90	

Aufgabe 1 [10 Punkte](Ausgabe und Eingabe).

- (a) Betrachte folgende Variablendeklarationen.

```
float x[] = {1, 2, 3};  
char y = '4';
```

Gebe beide Variablenwerte mit Hilfe der `printf` Funktion auf der Konsole aus.

- (b) Wir betrachten folgende Variablendeklaration.

```
char* x[] = {"Ha", "llo"};
```

Gebe den Inhalt des Arrays `x` mit Hilfe der `printf` Funktion auf der Konsole aus.

- (c) In dem folgenden Programmtext hat sich ein Fehler eingeschlichen. Korrigiere den Fehler.

```
char a[] = "a";  
float b = 2.0;  
  
printf("%s %f", b, a);
```

- (d) Betrachte folgenden Programtext.

```
int i[3] = {2, 1, 0};  
  
printf("%d %d %d %d", *i, i[1], *(i+1)+1, *(&i[0] + 1));
```

Welche Ausgabe findet sich auf der Konsole?

(e) Gegeben ist folgender C++ Programmtext.

```
char x = "3.3";  
cout << "Hallo " << 'a' << x;
```

Gebe eine äquivalente Formulierung in C.

Aufgabe 2 [15 Punkte](Parameterübergabe, Objekte und Zeiger).

(a) Gegeben ist folgende Inkrementierfunktion.

```
int inc(int x) { return x+1; }
```

Implementiere eine semantisch äquivalente Funktion mit Prototyp `void inc2(int, int*)`. Anstatt einem Rückgabewert wird eine Referenz (Zeiger) übergeben.

(b) Ersetze die Programmzeile `int x = inc(2);` indem die Funktion `inc2` anstatt `inc` verwenden.

(c) Gegeben ist folgende Objektdeklaration.

```
class MyInt {  
public:  
    int* x;  
    MyInt(int y) { x = new int(y); }  
    ~MyInt() { delete x; }  
};
```

Betrachte folgende Variablendeclarationen.

```
MyInt i = MyInt(1);  
MyInt* i2 = new MyInt(2);
```

Beschreibe im Detail welche Variablen/Objekten auf dem Stack beziehungsweise Heap angelegt werden.

(d) Welchen Wert liefert der Ausdruck `*(i2).x`?

(e) Betrachte

```
void func1(MyInt* o) {  
    *o->x = 11;  
}
```

War oder falsch. Der Aufruf `func1(&i)` führt zum Absturz.

(f) Betrachte

```
void func2(MyInt o) {  
    *o.x = 11;  
}
```

Erkläre im Detail. (1) Was geschieht bei Aufruf von `func2(i)`? (2) Was geschieht nach dem Aufruf von `func2(i)`?

Aufgabe 3 [45 Punkte](Strings, Arrays und Zeiger).

- (a) Wieviele Elemente hat folgendes Array?

```
char a[] = "Softwareprojekt";
```

- (b) Gegeben ist folgende `replace` Funktion welche in dem String `x` jedes Zeichen beschrieben durch `x` durch das Zeichen beschrieben durch `y` ersetzt.

```
void replace(char* s, char x, char y) {  
    while(*s != '\0') {  
        if(*s == x) {  
            *s = y;  
        }  
        s++;  
    }  
}
```

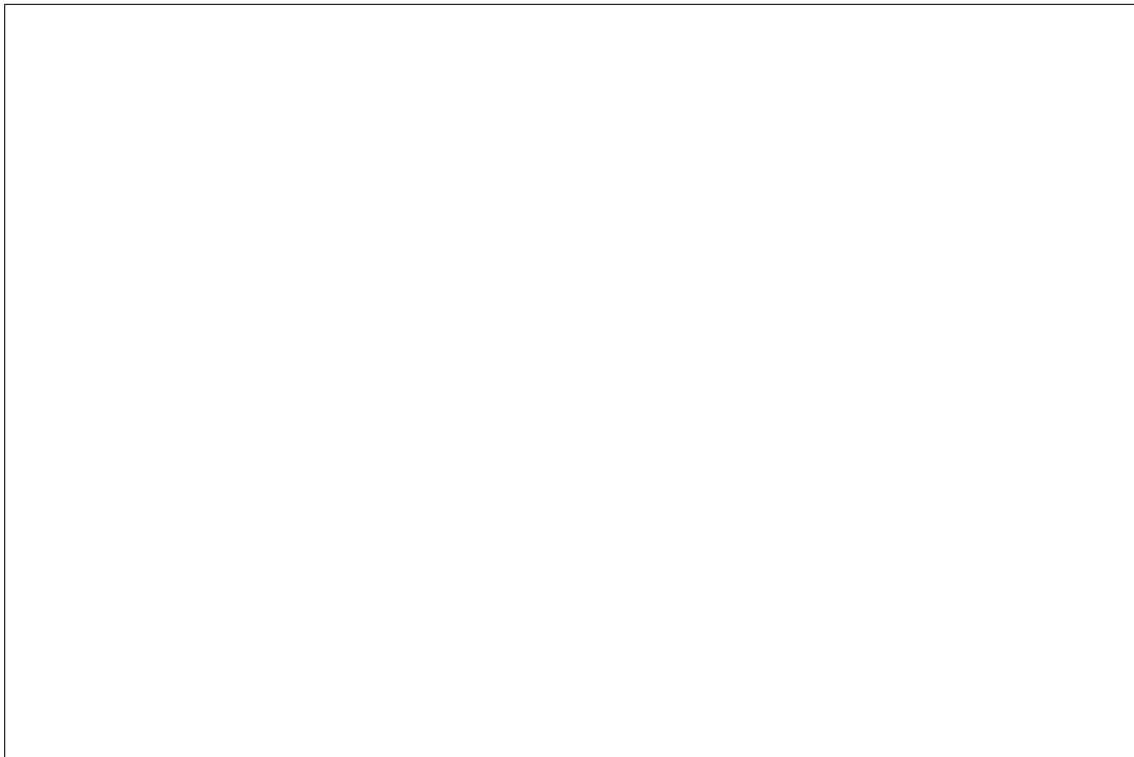
Gebe eine Alternative Implementierung in welcher keine Zeiger verwendet werden.

- (c) Implementiere eine Funktion mit dem Prototypen `int length(char*)` welche die Länge eines Strings berechnet.

- (d) Implementiere eine Variante von `replace` welche keine `for`, `do-while` oder `while` Schleife verwendet.



- (e) Implementiere eine weitere Variante von `replace` mit folgendem Funktionsprototypen `char* replace2(char*, char, char)`. Anstatt den bestehenden String zu überschreiben, wird ein neuer String erzeugt in welchem die Zeichen entsprechend ersetzt sind.



- (f) Implementiere eine Funktion mit dem Prototypen `int countCh(char* s, char x)` welche die Anzahl der Zeichen beschrieben durch `x` in dem String `s` berechnet.

- (g) Implementiere eine Funktion mit dem Prototypen `char* drop(char* s, char x)`. Eine neuer String wird erzeugt wobei in diesem String alle Zeichen beschrieben durch `x` aus dem String `s` entfernt wurden. Die relative Position aller anderer Zeichen bleibt gleich.

Aufgabe 4 [20 Punkte](Interpreter).

Gegeben ist folgender Programmtext.

```
typedef enum {
    INT = 0,
    PLUS = 1,
    MULT = 2
} Kind;

class Exp {
public:
    Exp() {};
    Kind k;
};

class IntExp : public Exp {
public:
    int val;
public:
    IntExp(int _val) { val = _val; k = INT; }
};

class PlusExp : public Exp {
public:
    Exp* e1;
    Exp* e2;
public:
    PlusExp(Exp* _e1, Exp* _e2) { e1 = _e1; e2 = _e2; k = PLUS; }
};

class MultExp : public Exp {
public:
    Exp* e1;
    Exp* e2;
public:
    MultExp(Exp* _e1, Exp* _e2) { e1 = _e1; e2 = _e2; k = MULT; }
};
```

Die Klassenhierarchie beschreibt arithmetische Ausdrücke. Z.B. $(1+2)*3$ kann dargestellt werden wie folgt

```
Exp* e = new MultExp(new PlusExp (new IntExp(1),
                                   new IntExp(2)),
                    new IntExp(3));
```

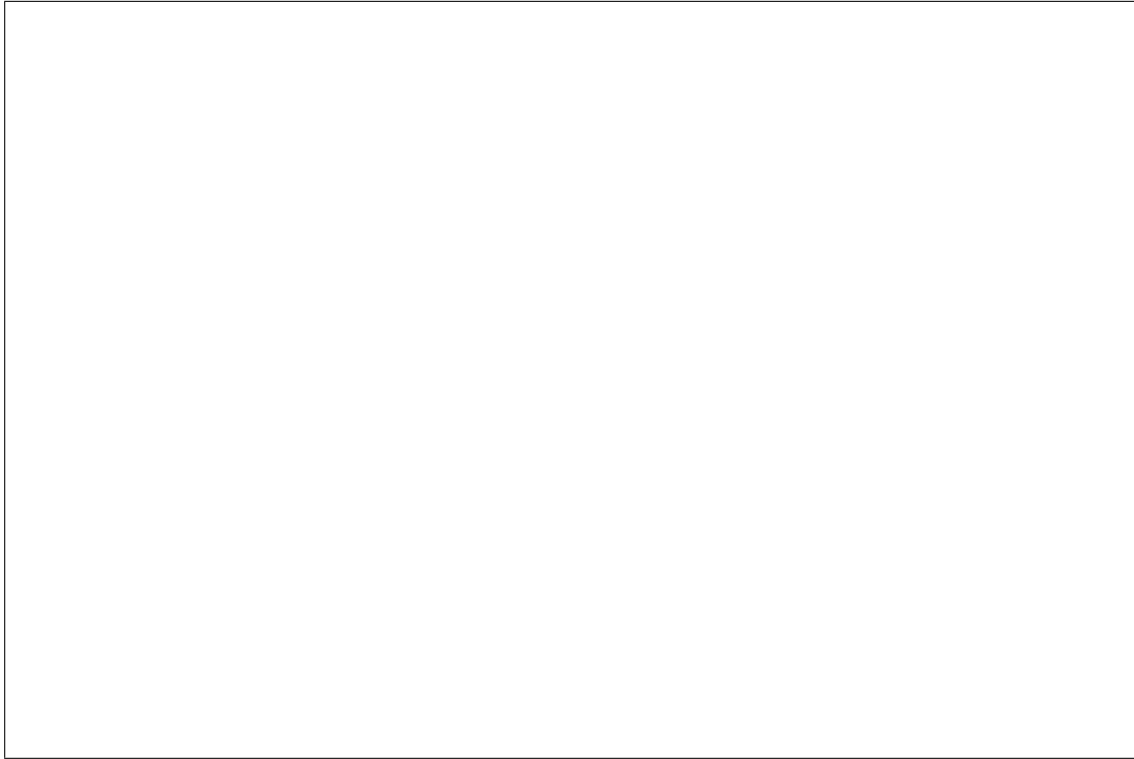
- (a) Betrachte den Ausdruck $1+2+3$. Gebe alle Darstellungen dieses Ausdrucks als ein C++ Objekt an.



- (b) Wir erweitern die Basisklasse `Exp` mit folgenden virtuellen Methoden.

```
class Exp {  
    public:  
        Exp() {};  
        virtual int eval() = 0;  
        virtual string reversePolish() = 0;  
};
```

Betrachte Methode `eval`. Ziel ist die Evaluation von arithmetischen Ausdrücken. Implementiere `eval` für die abgeleiteten Klassen.



- (c) Betrachte die Methode **reversePolish**. Ziel ist die Ausgabe eines arithmetischen Ausdrucks in der sogenannten umgekehrten polnischen Notation (“reverse polish notation”). Im Falle von

```
Exp* e = new PlusExp(new IntExp(1),  
                     new IntExp(2));
```

liefert **e->reversePolish()** den String "1 2 +". Sprich, zuerst kommen die Operanden dann der Operator.

Allgemein, betrachte den Ausdruck "**e1** + **e2**". Methode **reversePolish** wandelt zuerst **e1** und dann **e2** um in einen String. Beide Strings werden konkateniert. Dann wird der Operator + an diesen String angehängt.

Implementiere **reversePolish** für die abgeleiteten Klassen.



– Ende –

Musterloesung

(1a)

```
printf("%f %f %f", x[0], x[1], x[0]);  
printf("%d", y);
```

(1b)

```
printf("%s %s", x[0], x[1]);
```

(1c)

```
printf("%s %f", a, b);
```

(1d)

```
2 1 2 1
```

(1e)

Nicht gewertet, da Typo, sollte heissen

```
string x = "3.3";
```

(2a)

```
void inc2(int x, int* r) {  
    *r = inc(x);  
}
```

(2b)

```
int x;  
inc2(x, &x);
```

(2c)

Stack: i, *i2, i2

Heap: i.x, i2->x

(2d)

2

(2e)

Falsch

(2f)

Bei Aufruf von func2(i)

Kopie o.x = i.x

Nach Aufruf (bei verlassen der Funktion):

Stackfreigabe

=> Destruktur Aufruf fuer Objekt o

=> delete o.x

(entspricht delete i.x)

(3a)

 $15 + 1 = 16$

(3b)

```
void replace(char[] s, char x, char y) {  
    int i = 0;  
    while (s[i] != '\0') {  
        if(s[i] == x) {  
            s[i] = y;  
        }  
    }  
}
```

(3c)

```
int length (char* s) {  
    int i = 0;  
    while(*s != '\0') {  
        i++;  
        s++;  
    }  
    return i;  
}
```

(5d)

```
void replace(char* s, char x, char y) {  
    if(*s != '\0') {  
        if(*s == x) {
```

```
        *s == y;
    }
    s++;
    replace(s, x, y);
}
}
```

(5e)

```
char* replace2(char* s, char x, char y) {
    char* r = new char[length(s) + 1];
    char* t = r;
    while(*s != '\0') {
        if(*s == x) {
            *r = y;
        } else {
            *r = *s;
        }
        s++;
        r++;
    }
    *r = '\0';
    return t;
}
```

(5f)

```
int countCh(char *s, char x) {
    int i = 0;
    while(*s != '\0') {
        if(*s == x) {
            i++;
        }
        s++;
    }
    return i;
}
```

(5g)

```
char* drop(char* s, char x) {
    char* r = new char[length(s) - countCh(s,x) + 1];
    char* t = r;
    while(*s != '\0') {
        if(*s == x) {
            s++;
        } else {
            *t = *s;
            t++;
        }
    }
    *t = '\0';
    return r;
}
```

```
        *r = *s;
        s++;
        r++;
    }
}
*r = '\0';
return t;
}
```

(6a)

```
Exp* e1 = new PlusExp(new PlusExp(new IntExp(1),
                                   new IntExp(2)),
                     new IntExp(3));

Exp* e2 = New PlusExp(new IntExp(1),
                     new PlusExp(new IntExp(2),
                                   new IntExp(3)));
```

(6b)

```
int IntExp::eval() { return val; }

int PlusExp::eval() {
    return e1->eval() + e2->eval();
}

int MultExp::eval() {
    return e1->eval() * e2->eval();
}

string IntExp::reversePolish() {
    stringstream ss;
    ss << val;
    return ss.str();
}

string PlusExp::reversePolish() {
    string s;
    s.append(e1->reversePolish());
    s.append(e2->reversePolish());
    s.append("+");
    return s;
}

string MultExp::reversePolish() {
```



```
string s;  
s.append(e1->reversePolish());  
s.append(e2->reversePolish());  
s.append("*");  
return s;  
}
```