**7. Aufgabenblatt**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vorname** | **Nachname** | **Matrikelnummer** | **Fachrichtung** |
| Philipp | Huber | 63326 | Mabb |
| Marius | Grumer | 63284 | Mabb |

|  |  |
| --- | --- |
| **Punkte** | xx/yy |
| **Bestanden** | ja/nein |

*1. Aufgabe:*

In C++ stehen zur dynamischen Speicherreservierung die Operatoren ……………. und …………. Zur Verfügung.

**Lösung:** new und new[]

**Erklärung:** *Speicherplatz auf dem Heap (Haufen) kann mit den obigen Operatoren angefordert werden.*

*2. Aufgabe:*

Mit der Definition

**long\* ptr = new int;**

1. wird ein Objekt vom Typ *long* erzeugt
2. wird ein Objekt vom Typ *int* angelegt und implizit in *long* konvertiert
3. wird vom Compiler eine Fehlermeldung ausgegeben

**Lösung**: c

**Erklärung:** *getestet mit Fehlermeldung: Ein Wert vom Typ ""int \*"" kann nicht zum Initialisieren einer Entität vom Typ ""long \*"" verwendet werden.*

*3. Aufgabe:*

Wird für ein dynamisch reserviertes Objekt der *delete*-Operator nicht aufgerufen, so wird der dynamisch reservierte Speicher

1. nie freigegeben
2. automatisch freigegeben, wenn er nicht mehr verwendet wird
3. erst freigegeben, wenn das Programm beendet wird

**Lösung**: c

**Erklärung:** *Wenn der Dtor nicht aufgerufen wird das Objekt nicht zerstört. Es erst zerstört, sobald das Programm komplett beendet wurde (OS gibt Speicher frei).*

*4. Aufgabe:*

Angenommen, der Operator *new* wird mit einer Liste von Anfangswerten aufgerufen, um ein Objekt einer Klasse dynamisch anzulegen. Falls dann der Compiler keinen passenden Konstruktor findet, wird

1. eine Fehlermeldung ausgegeben
2. der Default-Konstruktor aufgerufen
3. eine Minimal-Version eines passenden Konstruktors erzeugt

**Lösung**: a

**Erklärung:** *Wenn als Parameter eine Liste von Werten Übergeben wird und kein passender Konstruktor gefunden wird, erfolgt die Ausgabe einer Fehlermeldung. Eine Minimal-Version des ctor wird nur bei einer Initialisierung ohne param erzeugt, falls kein ctor vorhanden ist.*

*5. Aufgabe:*

Ohne explizite Initialisierung wird für ein mit *new* erzeugtes Objekt der ……………………………aufgerufen.

**Lösung:** default ctor

*6. Aufgabe:*

Gegeben sei eine Klasse Test mit einer public-Methode display(). Nach der Definition

**Test\* ptr = new Test(1190.9);**

Kann dann die Methode display() wie folgt aufgerufen werden:

1. *new* -> *display*();
2. *ptr* -> *display*();
3. *ptr*. *display*();

**Lösung:** b

**Erklärung:** *Auf die Methoden und Member der Klasse Test kann von einem Pointer mit dem Pfeiloperator zugegriffen werden.*

*7. Aufgabe:*

Angenommen , eine Klasse LongArr besitzt ein dynamisches Element ptr, das einen Vektor variabler Länge adressiert. Dann muss der Destuktor zum freigeben des Speichers folgende Anweisung enthalten:

1. *delete* *ptr*[];
2. *delete*[] ptr;
3. *delete*[] \**ptr*;

**Lösung**: b

**Erklärung:** *um alle Elemente des Vektors zu zerstören muss das delete mit [] aufgerufen werden.*

*8. Aufgabe:*

Ein dynamisches Element einer Klasse wird als ………………………….. deklariert.

**Lösung**: Pointer

*9. Aufgabe:*

Angenommen, es soll eine Klasse *TestArr* entwickelt werden, die einen Vektor variabler Länge mit Elementen einer vorgegebener Klasse Test darstellt. Dann ist bei der Definition der Klasse

**Class TestArr**

**{private**

**int n; //Länge des Vektors**

**………..**

**};**

Ein dynamisches Element tPtr für den Vektor wie folgt zu deklarieren:

1. *Test* *tPtr*[n];
2. *Test* \**tPtr*;
3. *TestArr* \**tPtr*; ( = new TestArr;)

**Lösung**: c

**Erklärung:** *Um ein dynamisches Element der Klasse „TestArr“ mit dem Namen des Pointers „tPtr“ anzulegen ist die Syntax nach c) zu nehmen.*

*10. Aufgabe:*

Gegeben sei die Klasse:

**Class IntArr**

**{private**

**int \*ptrArr; //Zeiger auf dem Vektor**

**int len // Länge des Vektors**

**public**

**………..**

**};**

Bestimmen sie die Fehler in der Definition des

1. *Kopierkonstruktors*:

**IntArr::IntArr(const IntArr& src)**

**{**

**src.len=len;**

**src.ptrArr = new int[len];**

**for( int i=0;i<len;i++)**

**scr.ptrArr[i] = ptrArr[i];**

**}**

1. *Zuweisungsoperators*:

**IntArr& IntArr::operator=(const IntArr& src)**

**{**

**len = src.len;**

**ptrArr = new int[len];**

**for( int i=0;i<len;i++)**

**ptrArr[i] = src.ptrArr[i];**

**}**

**Lösung**:

**a. IntArr::IntArr(const IntArr& src)**

**{**

**len = src.len;**

**src.ptrArr\* = new int[len];**

**for( int i=0;i<len;i++)**

**ptrArr[i] = scr.ptrArr[i];**

**}**

**b.**

IntArr& IntArr::operator=(const IntArr& src)

{

if(this != &src) // auf selbstzuweisung ueberpruefen

{

len = src.len;

ptrArr = new int[len];

for (int i = 0; i < len; i++)

ptrArr[i] = src.ptrArr[i];

}

return \*this; // sich selbst zurückgeben

}