```
  1 //=============================================================================
  2 //        * Letsch Informatik *       www.LetsInfo.ch       CH-8636 Wald
  3 //          Beratung, Ausbildung und Realisation in Software-Engineering
  4 //=============================================================================
  5 // Project   : C++-Kurs @ Sultex Limited
  6 // Title     : Übung 1: Pointer, Referenzen und Memory-Layout: Loesung
  7 // Author    : Thomas Letsch
  8 // Tab-Width : 2
  9 /*///===========================================================================
 10 * Description:
 11 Programm um das Memory-Layout auf einer bestimmten Plattform und Umgebung
 12 zu untersuchen.
 13 Achtung: Keine Optimierungen beim Kompilieren!
 14 Sonst ist das Memory-Layout fast nicht nachvollziehbar.
 15 * History    :
 16 01.12.02: Initial Version.
 17 03.12.02: Fomatierung angepasst.
 18 12.12.03: ANSI-Header/using namespace std.
 19 26.09.05: Pointer/Referenzen hinzugefügt.
 20 /*///===========================================================================
 21 //       1         2         3         4         5         6         7         8
 22 //345678901234567890123456789012345678901234567890123456789012345678901234567890
 23 //=============================================================================
 24
 25 #include <iostream>
 26 #include <iomanip>
 27
 28 using namespace std;
 29
 30 // Function-Prototyps:
 31
 32 // Aufgabe 1:
 33 void funcPtr(string **pStr);
 34 void funcRef(string *&pStr);
 35
 36 // Aufgabe 2:
 37 int func1();
 38 int func2();
 39 int fac(int pN);
 40
 41 // dumpMemory():
 42 // Dumps Memory to stdout.
 43 // - pPtr:        Start-Adress from where to dump.
 44 // - pIdentStr:   Identify-String which is written first.
 45 // - pNrOfBytes:  Number of Bytes to be dumped.
 46 void dumpMemory(const void* pPtr, const char* pIdentStr = 0, int pNrOfBytes = 4);
 47
 48 // global Definitions:
 49 int        globalInt        = 1;
 50 const int  globalConstInt   = 2;
 51
 52 int main() {
 53
 54   cout << "Aufgabe 1:" << endl;
 55
 56   string* strPtr = 0;
 57   string** strPtrPtr = &strPtr;
 58   funcPtr(strPtrPtr);
 59   cout << "str = " << **strPtrPtr << endl;
 60   strPtr = 0;
 61   funcRef(strPtr);
 62   cout << "str = " << *strPtr << endl;
 63
 64
 65   cout << "\nAufgabe 2:" << endl;
 66
 67   int mainInt1 = 2147483647;     // 2^31 - 1
 68   int mainInt2 = -2;
 69   int* ip = new int;
 70   *ip = -1;
 71
 72   dumpMemory(&globalInt,       "globalInt",        4 );
 73   dumpMemory(&globalConstInt,  "globalConstInt",   4 );
 74   dumpMemory(&mainInt1,        "mainInt1",         sizeof(mainInt1) );
 75   dumpMemory(&mainInt2,        "mainInt2",         sizeof(mainInt2) );
 76   dumpMemory(&mainInt2,        "mainInt2 - 8",     8 );
 77   dumpMemory(&ip,              "ip",               sizeof(ip) );
 78   dumpMemory(ip,               "*ip",              sizeof(*ip) );
 79   dumpMemory((void*)main,      "main()");
 80   dumpMemory((void*)func1,     "func1()");
 81   dumpMemory((void*)func2,     "func2()");
 82   dumpMemory((void*)fac,       "fac()");
 83
 84   int ret = 0;
 85   ret = fac(5);
 86   cout << "fac(5) : " << ret << endl;
 87
 88 } // main()
 89
 90
 91
 92
 93 // Implementation Aufgabe 1:
 94
 95 void funcPtr(string** pStr) {
 96   *pStr = new string("funcPtr()");
 97 }
 98
 99
100 void funcRef(string*& pStr) {
101   pStr = new string("funcRef()");
102 }
103
104
105
```

```
106
107 // Implementation Aufgabe 2:
108
109 int func1() {
110   return 1;
111 }
112
113
114 int func2() {
115   return 2;
116 }
117
118
119 int fac(int pN) {
120   int newArg = pN - 1;   // Just for Demonstration of local Variable in Functions.
121   if (pN > 1) {
122     return pN * fac(newArg);
123   }
124   else {
125     dumpMemory(&newArg, "fac(5):newArg", 244); // Let's see the Stack right now !
126     return 1;                                  // With 244 Bytes we see enough.
127   }
128 }
129
130
131 void dumpMemory(const void* pPtr, const char* pIdentStr, int pNrOfBytes) {
132   if (pIdentStr != 0) {
133     cout << pIdentStr << ":" << endl;
134   }
135   unsigned char* cp = (unsigned char*)pPtr;
136   int words = (pNrOfBytes / 4) + (((pNrOfBytes % 4) > 0) ? 1 : 0);
137   int byteArr[] = {-1, -1, -1, -1};
138   for (int word = 0; word < words; word++) {      // for all Words
139     cout << setw(13) << setfill(' ') << static_cast<void*>(cp) << " : ";
140     for (int byte = 0; byte < 4; byte++) { // for all Bytes in Word
141       // The Bit-Mask: 128dec, left-most bit set to 1:
142       unsigned char mask = 0x80;
143       for (int bitpos = 0; bitpos < 8; bitpos++) { // for all 8 Bit-Positions
144         if (mask & *cp)                            // Mask BIT-AND actual Byte
145           cout << "1";
146         else
147           cout << "0";
148         mask = mask >> 1;
149       }
150       byteArr[byte] = *cp;                         // save Value for Hex print
151       cout << " ";
152       if ((word * 4) + (byte+1) == pNrOfBytes) {
153         break;
154       }
155       cp++;                                        // Increment Pointer by one Byte
156     } // for (int byte = 0; byte < 4; byte++)
157     // Append now Hex-Values:
158     cout << "   ";
159     for (int byte = 0; byte < 4; byte++) {
160       if (byteArr[byte] != -1) {
161         cout << " " << hex << setw(2) << setfill('0') << byteArr[byte];
162       }
163     }
164     for (int byte = 0; byte < 4; byte++) {
165       byteArr[byte] = -1;
166     }
167     cout << endl;
168   } // for (int word = 0; word <= words; word++)
169   cout << dec << endl;
170 } // dumpMemory()
171
```



```
172 /* Session-Log:
173
174 PC/Win200 mit Cygwin/Gnu:
175 ========================
176
177 $ uname -a
178 CYGWIN_NT-5.0 DONNA 1.3.12(0.54/3/2) 2002-07-06 02:16 i686 unknown
179 $
180 $ g++ MemoryLayout.cpp -o memlayout.exe
181 $
182 $ memlayout.exe
183 Aufgabe 1:
184 str = funcPtr()
185 str = funcRef()
186
187 Aufgabe 2:
188 globalInt:
189 0x413010 : 00000001 00000000 00000000 00000000  01 00 00 00
190
191 globalConstInt:
192 0x4015a4 : 00000010 00000000 00000000 00000000  02 00 00 00
193
194 mainInt1:
195 0x22fedc : 11111111 11111111 11111111 01111111  ff ff ff 7f
196
197 mainInt2:
198 0x22fed8 : 11111110 11111111 11111111 11111111  fe ff ff ff
199
200 mainInt2 - 8:
201 0x22fed8 : 11111110 11111111 11111111 11111111  fe ff ff ff
202 0x22fedc : 11111111 11111111 11111111 01111111  ff ff ff 7f
203
204 ip:
205 0x22fed4 : 01000000 00000011 00000001 00001010  40 03 01 0a
206
207 *ip:
208 0xa010340 : 11111111 11111111 11111111 11111111  ff ff ff ff
209
210 main():
211 0x4010a8 : 01010101 10001001 11100101 10000011  55 89 e5 83
212
213 func1():
214 0x401230 : 01010101 10001001 11100101 10111000  55 89 e5 b8
215
216 func2():
217 0x401240 : 01010101 10001001 11100101 10111000  55 89 e5 b8
218
219 fac():
220 0x401260 : 01010101 10001001 11100101 10000011  55 89 e5 83
221
```

```
222 fac(5):newArg:
223 0x22fdec : 00000000 00000000 00000000 00000000  00 00 00 00 | newArg : 0          |
224 0x22fdf0 : 00100000 11111110 00100010 00000000  20 fe 22 00 |                     |
225 0x22fdf4 : 01110111 00010010 01000000 00000000  77 12 40 00 | PC: fac(): 0x401260 |
226 0x22fdf8 : 00000001 00000000 00000000 00000000  01 00 00 00 | pN : 1              |
227 0x22fdfc : 01000100 00110000 01000001 00000000  44 30 41 00 |                     |
228 0x22fe00 : 00110000 11111110 00100010 00000000  30 fe 22 00 |                     |
229 0x22fe04 : 01010100 00010011 01000001 00000000  54 13 41 00 |                     |
230 0x22fe08 : 01000000 00110000 01000001 00000000  40 30 41 00 |                     |
231 0x22fe0c : 00000001 00000000 00000000 00000000  01 00 00 00 |                     |
232 0x22fe10 : 10000011 00000000 00000000 00000000  83 00 00 00 +- fac(1) ------------+
233 0x22fe14 : 00000001 00000000 00000000 00000000  01 00 00 00 |                     |
234 0x22fe18 : 00000001 00000000 00000000 00000000  01 00 00 00 |                     |
235 0x22fe1c : 00000001 00000000 00000000 00000000  01 00 00 00 | newArg : 1          |
236 0x22fe20 : 01010000 11111110 00100010 00000000  50 fe 22 00 |                     |
237 0x22fe24 : 01110111 00010010 01000000 00000000  77 12 40 00 | PC: fac(): 0x401260 |
238 0x22fe28 : 00000010 00000000 00000000 00000000  02 00 00 00 | pN : 2              |
239 0x22fe2c : 01000000 00110000 01000001 00000000  40 30 41 00 |                     |
240 0x22fe30 : 10110000 11111110 00100010 00000000  b0 fe 22 00 |                     |
241 0x22fe34 : 10010001 00010101 01000000 00000000  91 15 40 00 |                     |
242 0x22fe38 : 01000000 00110000 01000001 00000000  40 30 41 00 |                     |
243 0x22fe3c : 10110100 00101110 01000000 00000000  b4 2e 40 00 |                     |
244 0x22fe40 : 00000000 00000000 00000000 00000000  00 00 00 00 +- fac(2) ------------+
245 0x22fe44 : 00000000 00000000 00000000 00000000  00 00 00 00 |                     |
246 0x22fe48 : 00000000 00000000 00000000 00000000  00 00 00 00 |                     |
247 0x22fe4c : 00000010 00000000 00000000 00000000  02 00 00 00 | newArg : 2          |
248 0x22fe50 : 10000000 11111110 00100010 00000000  80 fe 22 00 |                     |
249 0x22fe54 : 01110111 00010010 01000000 00000000  77 12 40 00 | PC: fac(): 0x401260 |
250 0x22fe58 : 00000011 00000000 00000000 00000000  03 00 00 00 | pN : 3              |
251 0x22fe5c : 00000000 00000000 00000000 00000000  00 00 00 00 |                     |
252 0x22fe60 : 00000000 00000000 00000000 00000000  00 00 00 00 |                     |
253 0x22fe64 : 00000000 00000000 00000000 00000000  00 00 00 00 |                     |
254 0x22fe68 : 00011000 00110111 01000001 00000000  18 37 41 00 |                     |
255 0x22fe6c : 00000100 00000000 00000000 00000000  04 00 00 00 |                     |
256 0x22fe70 : 10101000 00010110 01000000 00000000  a8 16 40 00 +- fac(3) ------------+
257 0x22fe74 : 00000010 00000000 00000000 00000000  02 00 00 00 |                     |
258 0x22fe78 : 01111100 00010110 01000000 00000000  7c 16 40 00 |                     |
259 0x22fe7c : 00000011 00000000 00000000 00000000  03 00 00 00 | newArg : 3          |
260 0x22fe80 : 10110000 11111110 00100010 00000000  b0 fe 22 00 |                     |
261 0x22fe84 : 01110111 00010010 01000000 00000000  77 12 40 00 | PC: fac(): 0x401260 |
262 0x22fe88 : 00000100 00000000 00000000 00000000  04 00 00 00 | pN : 4              |
263 0x22fe8c : 00000001 00000000 00000000 00000000  01 00 00 00 |                     |
264 0x22fe90 : 11111111 11111111 11111111 11111111  ff ff ff ff |                     |
265 0x22fe94 : 11111111 11111111 11111111 11111111  ff ff ff ff |                     |
266 0x22fe98 : 11111111 11111111 11111111 11111111  ff ff ff ff |                     |
267 0x22fe9c : 11111111 11111111 11111111 11111111  ff ff ff ff |                     |
268 0x22fea0 : 10011000 11111101 00100010 00000000  98 fd 22 00 +- fac(4) ------------+
269 0x22fea4 : 11000100 00000000 00000000 00000000  c4 00 00 00 |                     |
270 0x22fea8 : 00000001 00000000 00000000 00000000  01 00 00 00 |                     |
271 0x22feac : 00000100 00000000 00000000 00000000  04 00 00 00 | newArg : 4          |
272 0x22feb0 : 11100000 11111110 00100010 00000000  e0 fe 22 00 |                     |
273 0x22feb4 : 11100101 00010001 01000000 00000000  e5 11 40 00 | PC: main(): 0x4010a8|
274 0x22feb8 : 00000101 00000000 00000000 00000000  05 00 00 00 | pN : 5              |
275 0x22febc : 10010110 00010000 01000000 00000000  96 10 40 00 |                     |
276 0x22fec0 : 00000100 00000000 00000000 00000000  04 00 00 00 |                     |
277 0x22fec4 : 10110011 00010000 01000000 00000000  b3 10 40 00 |                     |
278 0x22fec8 : 10110100 00010011 01101000 01100001  b4 13 68 61 |                     |
279 0x22fecc : 00101111 00000000 00000000 00000000  2f 00 00 00 |                     |
280 0x22fed0 : 00000000 00000000 00000000 00000000  00 00 00 00 + - fac(5): `return` -+
281 0x22fed4 : 01000000 00000011 00000001 00001010  40 03 01 0a ip
282 0x22fed8 : 11111110 11111111 11111111 11111111  fe ff ff ff mainInt2
283 0x22fedc : 11111111 11111111 11111111 01111111  ff ff ff 7f mainInt1
284
285 fac(5) : 120
286 $
287
288 Hinweis: Beschreibungen bei fac(5) sind nachträglich hinzugefügt.
289
```



```
290 Memory-Layout:
291 =============
292
293
294
295
296
297 Bereich                                    Adresse      Datenelement-Name
298                                  Richtung
299 -----------------------------------+-----------------------------------
300                                    0x000000
301
302                                  ^
303                                  ^
304                                  ^
305                                  ^
306                                  ^
307                                  ^    0x0022fed4   ip
308                                  ^    0x0022fed8   mainInt2
309                                  ^    0x0022fedc   mainInt1
310                                  ^
311 Stack: ----------------------------+-------------------------------
312
313 Code-Segment: --------------------+ -------------------------------
314                                  v
315                                  v    0x004010a8   main()
316                                  v    0x00401230   func1()
317                                  v    0x00401240   func2()
318                                  v    0x00401260   fac()
319                                  v
320                                  v
321 Daten-Segment: Konstanten: ---------+ -------------------------------
322                                  v
323                                  v    0x004015a4   globalConstInt
324                                  v
325                                  v
326 Daten-Segment: globale Variablen: --+-------------------------------
327                                  v
328                                  v    0x00413010   globalInt
329                                  v
330                                  v
331 Dynamischer Speicher / Heap --------+-------------------------------
332                                  v
333                                  v    0x0a010340   ip
334                                  v
335                                  v
336                                  v
337                                  v
338                                  v
339                                  v
340
341                                    0xffffff
342 -----------------------------------+-------------------------------
343
344
345
346
347 Richtung:
348 ^: Adressen sind abnehmend
349 v: Adressen sind zunehmend
350
351 */
```