

Lecture 8: C language

- History of C
- Structure of a C program
- C data types
- Variable declaration and scope
- C operators
- Loops and iterations
- Pointers
- Structures in C
- C and assembly language

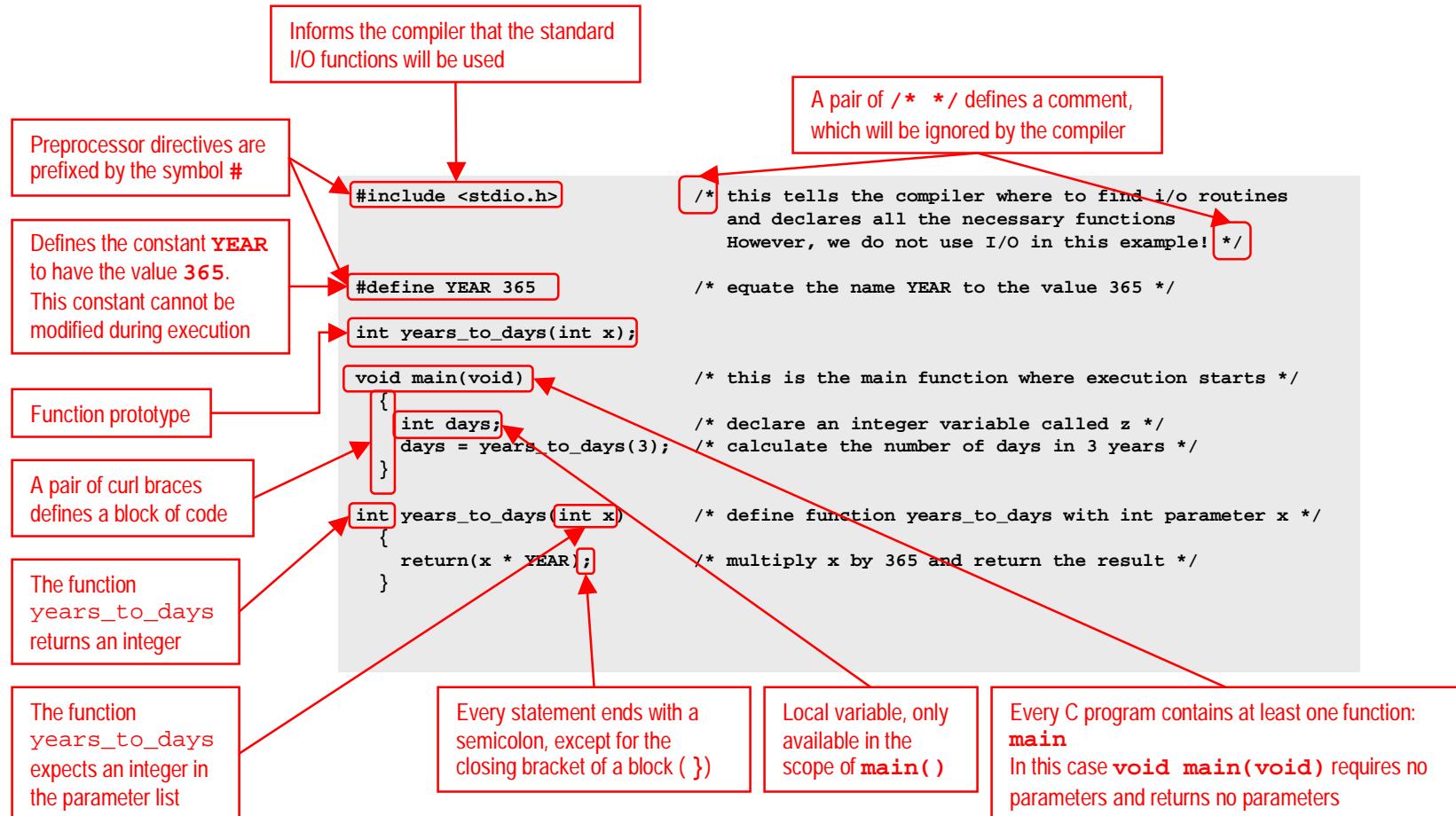


History of C

- Developed in 1972 by Dennis Ritchie on a DEC PDP-11 at Bell Systems Lab as a *system development language*
 - Derived from the language B of Ken Thompson, which itself was based on BCPL, developed by Martin Richards
- For many years the *de-facto* C standard was the version provided with Unix System V
 - The C Programming Language, Brian Kernighan and Dennis Ritchie, Prentice-Hall 1978
- In 1983 ANSI creates a group to begin the standardization of C
 - ANSI C is finalized in 1989, and ISO adopts it in 1990



Structure of a C program



C data types

■ Four basic data types

- `char`: character
- `int`: integer
- `float`: real or floating point
- `double`: double precision float

■ Four modifiers

- `signed`
- `unsigned`
- `long`
- `short`

■ Four storage classes

- `auto`: variable is not required outside its block (the default)
- `register`: the variable will be allocated on a CPU register
- `static`: allows a local variable to retain its previous value upon reentry
- `extern`: global variable declared in another file

■ Additionally, C supports

- the null data type: `void`
- Any user-defined types

Type	Width (bits)	Minimum range
<code>char</code>	8	-127 to 127
<code>unsigned char</code>	8	0 to 255
<code>signed char</code>	8	-127 to 127
<code>int</code>	16	-32,767 to 32,767
<code>unsigned int</code>	16	0 to 65,535
<code>signed int</code>	16	Same as int
<code>short int</code>	16	Same as int
<code>unsigned short int</code>	8	0 to 65,535
<code>signed short int</code>	8	Same as short int
<code>long int</code>	32	-2,147,483,647 to 2,147,483,647
<code>signed long int</code>	32	--2,147,483,647 to 2,147,483,647
<code>unsigned long int</code>	32	0 to 4,294,967,295
<code>float</code>	32	Six-digit precision
<code>double</code>	64	Ten-digit precision
<code>long double</code>	128	Ten-digit precision



Variable declaration and scope

- **Variables MUST be declared before they are used**
 - Any declaration MUST precede the first statement in a block
- **Variables declared inside a block are local to that block**
 - They cannot be accessed from outside the block
- **Variables can be initialized when they are declared or afterwards**

```
int i;                      /* Integer i is global to the entire program
                                and is visible to everything from this point */
void function_1(void)        /* A function with no parameters */
{
    int k;                  /* Integer k is local to function_1 */
    {
        int q;              /* Integer q exists only in this block */
        int j;              /* Integer j is local and not the same as j in main */

    }
}
void main(void)
{
    int j;                  /* Integer j is local to this block within function main */
}

}                          /* This is the point at which integer j ceases to exist */
```



C operators

Type	Operator	Action
Arithmetic	-	Subtraction
	+	Addition
	*	Multiplication
	/	Division
	%	Modulus
	--	Decrement (by 1)
	++	Increment (by 1)
	+=	Increment ($a+=b$ means $a=a+b$)
	-=	Decrement ($a-=b$ means $a=a-b$)
Relational	>	Greater than
	>=	Greater than or equal to
	<	Less than
	<=	Less than or equal to
	==	Equal to
	!=	Different from
Logic	&&	AND
		OR
	!	NOT
Bit-wise	&	AND
		OR
	^	XOR
	~	NOT
	>>	Right shift
	<<	Left shift
Miscellaneous	?	Ternary ($y=x>9?100:200$)
	& and *	Pointer operators
	sizeof	Width of a datatype (in bytes)
	. and ->	Access to structures
	[]	Access to arrays

Precedence	Operator
Most ↑ Least	() [] -> . ! ~ ++ -- - (cast) * & sizeof / % << >> < <= > >= == != & && ? = += -= *= /= `



Loops and iterations

- In C any expression different than ZERO is TRUE, including negative numbers, strings, ...
- C provides the following constructs

if-else

```
if (expr2) {  
    block2;  
} else if (expr3) {  
    block3;  
} else {  
    default_block;  
}
```

while, do-while

```
while (expression) {  
    block;  
}  
  
do {  
    block;  
} while (expression);
```

goto

```
goto label;  
block1;  
label:  
block2;
```

for

```
for (initialization;condition;increment) {  
    block;  
}  
  
for (;;) {  
    block;  
    if (expr)  
        break;  
}
```

switch-case

```
switch (expression) {  
    case constant1:  
        block1;  
        break;  
    case constant2:  
        block2;  
        break;  
    default:  
        block_default;  
}
```



Pointers (I)

- A pointer is a variable that stores a memory address
- A pointer must be declared and initialized before it can be used

```
void main() {
    int a=10;
    int *p;
    p = &a;
    *p = 20; /* 'a' contains the value 20*/
}
```

■ Pointers and arrays are closely related

- the name of the array serves as a pointer to its first element
- the first element has index 0
- array elements can be addressed using brackets or pointer arithmetic

```
void main() {
    int array[5]={1,2,3,4,5};
    int value, *p;

    p = &array[0]; /* these two expressions */
    p = array;      /* are equivalent */

    array[2];        /* both expressions point to */
    *(p+2);         /* the 3rd element in array */
}
```

■ Strings of characters and arrays are closely related

- A string is an array of characters followed by the '\0' null character

```
void main() {
    char *string1="hello"
    char string2[6]={'h','e','l','l','o','\0'};
    char *p_string;

    p_string = string1;

    printf("%s\n",string1); /* these expressions */
    printf("%s\n",string2); /* will produce the */
    printf("%s\n",p_string); /* same result */
}
```



Pointers (II)

■ Pointers can point to C functions

- The pointer will point to the memory address that stores the first instruction of the function
- Our knowledge of assembly language makes this idea easier to understand, doesn't it?

```
#include <stdio.h>
#include <string.h>

void my_strcmp(char *a, char *b, int (*ptr)()) {
    if ( !(*ptr)(a,b) ) printf("EQUAL");
    else printf("DIFFERENT");
}

void main() {
    char c1[80], c2[80];
    int (*p)();
    p = strcmp; /* p points to the function strcmp() */
    gets(c1); /* get the strings */
    gets(c2); /* from the keyboard */
    my_strcmp(c1,c2,p);
}
```

■ Pointers and dynamic memory allocation

- Sometimes the length of an array is unknown at compilation time
- Using pointers and the `malloc()` family of instructions we can allocate memory at run-time

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void main(void) {
    char *c;
    int i;

    c = malloc(80); /* allocate 80 characters */
    if (c==NULL) { /* error handling */
        printf("error allocating memory\n");
        exit(1);
    }

    gets(c); /* get a string from keyboard*/
    for (i=strlen(c)-1;t>=0;t--) {
        putchar(c[i]); /* display each element of string */
    }
    free(c); /* deallocate memory !!! */
}
```



Structures in C

- C allows definition of non-homogeneous data types with multiple fields, called structures

```
struct struct_def {  
    type field1;  
    type field2;  
...  
    type fieldn;  
};  
  
struct struct_def struct_instance;
```

-
- Fields in a structures can be accessed using ‘dot’ notation or ‘arrow’ notation

```
struct employee {  
    char      family_name[30];  
    char      first_name[30];  
    long int  stipend;  
};  
struct employee gra, *ptr;  
  
gra.stipend    = 1000;    /* these two expressions */  
ptr->stipend  = 1000;    /* are equivalent */
```

-
- Structure fields can be arrays and we can define arrays of structures

```
struct employee staff[200];  
  
strcpy(staff[0].family_name, "Doe");  
strcpy(staff[0].first_name, "John");  
staff[0].stipend = 10000;
```



Cross-compilation

- To generate the assembly code from a C program you can use the cross-compiler provided in the CD-ROM

- Cross-compiler is located in:

- CD_drive:\C\FILES\I2DEMO\ITOOLS\X\c68332.exe**

- To cross-compile

- **c68332 filename.c -no -i**

- where:

- **filename.c** is the name of the source file
 - **-no** is an option that suppresses compiler optimization
 - **-i** is an option that control the format of the output

- Read the “**CD_drive:\C\readme.txt**” file for more information



C and assembly language: example 1

```

int adder(int x, int y) {
    return x + y;
}

void main (void) {
    register int a, b, c;
    a = 1; b = 2;
    c = adder(a, b);
}

```

SP,A6

OLD_A6	OLD_A6
OLD_A6	OLD_A6
RET_ADD	RET_ADD
RET_ADD	RET_ADD
D7	D7
D6	D6

8(A6)
10(A6)

```

* Feb 7 1999 18:46:00
* bc sid : @(#)bc68000.PL 5.133.1.2
* options : -no -p -t 68332 -nv
* cpf sid : @(#)cpf.PL 6.66.1.4
*1     int adder(int x, int y) {

* Parameter x is at 8(A6)
* Parameter y is at 10(A6)

        000000      4e560000      _adder
                                         __P1      SECTION      S_adder,, "code"
                                         LINK      EQU      _adder
                                         MOVE      A6,#0
                                         ADD      $000004
                                         MOVE      8(A6),D1
                                         MOVE      10(A6),D1
                                         MOVE      D1,D0
                                         UNLK      A6
                                         RTS      A6

*2     return x + y;
        000000 + __P1 322e0008      XDEF      _adder
        000004 + __P1 d26e000a      XREF      __main
        000008 + __P1 3001          XREF      __main
*3   }
        00000e      4e5e          XDEF      _main
        000010      4e75          XREF      __main
                                         EQU      $000012
                                         MOVEQ.L #1,D7
                                         MOVEQ.L #2,D6
                                         MOVE      D6,-(A7)
                                         MOVE      D7,-(A7)
                                         JSR      _adder
                                         MOVE      D0,D5
                                         ADDQ.L  #4,A7
                                         RTS      A7

*4
*5     void main (void) {
                                         XREF      __main
                                         XDEF      _main
                                         EQU      $000012
                                         RET      A7

*6     register int a, b, c;
*7     a = 1; b = 2;
        000000 + __P2 7e01          XDEF      _main
        000002 + __P2 7c02          XREF      __main
                                         MOVEQ.L #1,D7
                                         MOVEQ.L #2,D6
                                         MOVE      D6,-(A7)
                                         MOVE      D7,-(A7)
                                         JSR      _adder
                                         MOVE      D0,D5
                                         ADDQ.L  #4,A7
                                         RTS      A7

*8     c = adder(a, b);
        000004 + __P2 3f06          XDEF      _main
        000006 + __P2 3f07          XREF      __main
        000008 + __P2 4eb9          XREF      __main
        00000e + __P2 3a00          XREF      __main
        000022      588f          XREF      __main
                                         MOVE      D6,-(A7)
                                         MOVE      D7,-(A7)
                                         JSR      _adder
                                         MOVE      D0,D5
                                         ADDQ.L  #4,A7
                                         RTS      A7

*9   }
        000024      4e75          XREF      __main
                                         END      _dgroup
                                         data

```



C and assembly language: example 2

```

void swap (int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
void main (void)
{
    int x = 2, y = 3;
    swap (x, y);
}

```

SP	TMP
A6	00 02
	OLD_A6 OLD_A6
	OLD_A6 OLD_A6
	RET_ADD RET_ADD
	RET_ADD RET_ADD
A	00 02 (03)
B	00 03 (02)
Y	00 03
X	00 02
	OLD_A6 OLD_A6
	OLD_A6 OLD_A6

```

*1      void swap (int a, int b)                                SECTION      S_swap,, "code"
* Parameter a is at 8(A6)                                         XDEF        _swap
* Parameter b is at 10(A6)                                         XDEF        _swap
* Variable temp is at -2(A6)                                         XDEF        _swap
*2      {                                                       SECTION      S_swap,, "code"
*3          int temp;                                              XREF        _main
*4          temp = a;                                              XREF        _main
*5          a = b;                                                 MOVE       8(A6),-2(A6)
*6          b = temp;                                             MOVE       10(A6),8(A6)
*7      }                                                       SECTION      S_swap,, "code"
*8      void main (void)                                         XREF        _main
* Function size = 26
*9      {                                                       SECTION      S_main,, "code"
*10         int x = 2, y = 3;                                       XREF        _main
*11         00000 + __P1   3d6e0008ffff                         MOVE       #2,-2(A6)
*12         00006 + __P1   3d6e000a0008                         MOVE       #3,-4(A6)
*13         swap (x, y);                                         MOVE       #3,-(A7)
*14         0000c + __P1   3f3c0003                            MOVE       #2,-(A7)
*15         00010 + __P1   3f3c0002                            MOVE       _swap
*16         00014 + __P1   4eb9______                           JSR        _swap
*17         00038           4e5e                               UNLK      A6
*18         0003a           4e75                               RTS       A6
* Function size = 34
* bytes of code = 60
* bytes of idata = 0
* bytes of udata = 0
* bytes of sdata = 0
*19
*20
*21
*22
*23
*24
*25
*26
*27
*28
*29
*30
*31
*32
*33
*34
*35
*36
*37
*38
*39
*40
*41
*42
*43
*44
*45
*46
*47
*48
*49
*50
*51
*52
*53
*54
*55
*56
*57
*58
*59
*60
*61
*62
*63
*64
*65
*66
*67
*68
*69
*70
*71
*72
*73
*74
*75
*76
*77
*78
*79
*80
*81
*82
*83
*84
*85
*86
*87
*88
*89
*90
*91
*92
*93
*94
*95
*96
*97
*98
*99
*100
*101
*102
*103
*104
*105
*106
*107
*108
*109
*110
*111
*112
*113
*114
*115
*116
*117
*118
*119
*120
*121
*122
*123
*124
*125
*126
*127
*128
*129
*130
*131
*132
*133
*134
*135
*136
*137
*138
*139
*140
*141
*142
*143
*144
*145
*146
*147
*148
*149
*150
*151
*152
*153
*154
*155
*156
*157
*158
*159
*160
*161
*162
*163
*164
*165
*166
*167
*168
*169
*170
*171
*172
*173
*174
*175
*176
*177
*178
*179
*180
*181
*182
*183
*184
*185
*186
*187
*188
*189
*190
*191
*192
*193
*194
*195
*196
*197
*198
*199
*200
*201
*202
*203
*204
*205
*206
*207
*208
*209
*210
*211
*212
*213
*214
*215
*216
*217
*218
*219
*220
*221
*222
*223
*224
*225
*226
*227
*228
*229
*230
*231
*232
*233
*234
*235
*236
*237
*238
*239
*240
*241
*242
*243
*244
*245
*246
*247
*248
*249
*250
*251
*252
*253
*254
*255
*256
*257
*258
*259
*260
*261
*262
*263
*264
*265
*266
*267
*268
*269
*270
*271
*272
*273
*274
*275
*276
*277
*278
*279
*280
*281
*282
*283
*284
*285
*286
*287
*288
*289
*290
*291
*292
*293
*294
*295
*296
*297
*298
*299
*300
*311
*312
*313
*314
*315
*316
*317
*318
*319
*320
*321
*322
*323
*324
*325
*326
*327
*328
*329
*330
*331
*332
*333
*334
*335
*336
*337
*338
*339
*340
*341
*342
*343
*344
*345
*346
*347
*348
*349
*350
*351
*352
*353
*354
*355
*356
*357
*358
*359
*360
*361
*362
*363
*364
*365
*366
*367
*368
*369
*370
*371
*372
*373
*374
*375
*376
*377
*378
*379
*380
*381
*382
*383
*384
*385
*386
*387
*388
*389
*390
*391
*392
*393
*394
*395
*396
*397
*398
*399
*400
*401
*402
*403
*404
*405
*406
*407
*408
*409
*410
*411
*412
*413
*414
*415
*416
*417
*418
*419
*420
*421
*422
*423
*424
*425
*426
*427
*428
*429
*430
*431
*432
*433
*434
*435
*436
*437
*438
*439
*440
*441
*442
*443
*444
*445
*446
*447
*448
*449
*450
*451
*452
*453
*454
*455
*456
*457
*458
*459
*460
*461
*462
*463
*464
*465
*466
*467
*468
*469
*470
*471
*472
*473
*474
*475
*476
*477
*478
*479
*480
*481
*482
*483
*484
*485
*486
*487
*488
*489
*490
*491
*492
*493
*494
*495
*496
*497
*498
*499
*500
*501
*502
*503
*504
*505
*506
*507
*508
*509
*510
*511
*512
*513
*514
*515
*516
*517
*518
*519
*520
*521
*522
*523
*524
*525
*526
*527
*528
*529
*530
*531
*532
*533
*534
*535
*536
*537
*538
*539
*540
*541
*542
*543
*544
*545
*546
*547
*548
*549
*550
*551
*552
*553
*554
*555
*556
*557
*558
*559
*5510
*5511
*5512
*5513
*5514
*5515
*5516
*5517
*5518
*5519
*5520
*5521
*5522
*5523
*5524
*5525
*5526
*5527
*5528
*5529
*5530
*5531
*5532
*5533
*5534
*5535
*5536
*5537
*5538
*5539
*5540
*5541
*5542
*5543
*5544
*5545
*5546
*5547
*5548
*5549
*5550
*5551
*5552
*5553
*5554
*5555
*5556
*5557
*5558
*5559
*55510
*55511
*55512
*55513
*55514
*55515
*55516
*55517
*55518
*55519
*55520
*55521
*55522
*55523
*55524
*55525
*55526
*55527
*55528
*55529
*55530
*55531
*55532
*55533
*55534
*55535
*55536
*55537
*55538
*55539
*55540
*55541
*55542
*55543
*55544
*55545
*55546
*55547
*55548
*55549
*55550
*55551
*55552
*55553
*55554
*55555
*55556
*55557
*55558
*55559
*55560
*55561
*55562
*55563
*55564
*55565
*55566
*55567
*55568
*55569
*55570
*55571
*55572
*55573
*55574
*55575
*55576
*55577
*55578
*55579
*55580
*55581
*55582
*55583
*55584
*55585
*55586
*55587
*55588
*55589
*55590
*55591
*55592
*55593
*55594
*55595
*55596
*55597
*55598
*55599
*555100
*555101
*555102
*555103
*555104
*555105
*555106
*555107
*555108
*555109
*555110
*555111
*555112
*555113
*555114
*555115
*555116
*555117
*555118
*555119
*555120
*555121
*555122
*555123
*555124
*555125
*555126
*555127
*555128
*555129
*555130
*555131
*555132
*555133
*555134
*555135
*555136
*555137
*555138
*555139
*555140
*555141
*555142
*555143
*555144
*555145
*555146
*555147
*555148
*555149
*555150
*555151
*555152
*555153
*555154
*555155
*555156
*555157
*555158
*555159
*555160
*555161
*555162
*555163
*555164
*555165
*555166
*555167
*555168
*555169
*555170
*555171
*555172
*555173
*555174
*555175
*555176
*555177
*555178
*555179
*555180
*555181
*555182
*555183
*555184
*555185
*555186
*555187
*555188
*555189
*555190
*555191
*555192
*555193
*555194
*555195
*555196
*555197
*555198
*555199
*555200
*555201
*555202
*555203
*555204
*555205
*555206
*555207
*555208
*555209
*555210
*555211
*555212
*555213
*555214
*555215
*555216
*555217
*555218
*555219
*555220
*555221
*555222
*555223
*555224
*555225
*555226
*555227
*555228
*555229
*555230
*555231
*555232
*555233
*555234
*555235
*555236
*555237
*555238
*555239
*555240
*555241
*555242
*555243
*555244
*555245
*555246
*555247
*555248
*555249
*555250
*555251
*555252
*555253
*555254
*555255
*555256
*555257
*555258
*555259
*555260
*555261
*555262
*555263
*555264
*555265
*555266
*555267
*555268
*555269
*555270
*555271
*555272
*555273
*555274
*555275
*555276
*555277
*555278
*555279
*555280
*555281
*555282
*555283
*555284
*555285
*555286
*555287
*555288
*555289
*555290
*555291
*555292
*555293
*555294
*555295
*555296
*555297
*555298
*555299
*5552100
*5552101
*5552102
*5552103
*5552104
*5552105
*5552106
*5552107
*5552108
*5552109
*5552110
*5552111
*5552112
*5552113
*5552114
*5552115
*5552116
*5552117
*5552118
*5552119
*5552120
*5552121
*5552122
*5552123
*5552124
*5552125
*5552126
*5552127
*5552128
*5552129
*5552130
*5552131
*5552132
*5552133
*5552134
*5552135
*5552136
*5552137
*5552138
*5552139
*5552140
*5552141
*5552142
*5552143
*5552144
*5552145
*5552146
*5552147
*5552148
*5552149
*5552150
*5552151
*5552152
*5552153
*5552154
*5552155
*5552156
*5552157
*5552158
*5552159
*5552160
*5552161
*5552162
*5552163
*5552164
*5552165
*5552166
*5552167
*5552168
*5552169
*5552170
*5552171
*5552172
*5552173
*5552174
*5552175
*5552176
*5552177
*5552178
*5552179
*5552180
*5552181
*5552182
*5552183
*5552184
*5552185
*5552186
*5552187
*5552188
*5552189
*5552190
*5552191
*5552192
*5552193
*5552194
*5552195
*5552196
*5552197
*5552198
*5552199
*5552200
*5552201
*5552202
*5552203
*5552204
*5552205
*5552206
*5552207
*5552208
*5552209
*5552210
*5552211
*5552212
*5552213
*5552214
*5552215
*5552216
*5552217
*5552218
*5552219
*5552220
*5552221
*5552222
*5552223
*5552224
*5552225
*5552226
*5552227
*5552228
*5552229
*55522210
*55522211
*55522212
*55522213
*55522214
*55522215
*55522216
*55522217
*55522218
*55522219
*55522220
*55522221
*55522222
*55522223
*55522224
*55522225
*55522226
*55522227
*55522228
*55522229
*55522230
*55522231
*55522232
*55522233
*55522234
*55522235
*55522236
*55522237
*55522238
*55522239
*55522240
*55522241
*55522242
*55522243
*55522244
*55522245
*55522246
*55522247
*55522248
*55522249
*55522250
*55522251
*55522252
*55522253
*55522254
*55522255
*55522256
*55522257
*55522258
*55522259
*55522260
*55522261
*55522262
*55522263
*55522264
*55522265
*55522266
*55522267
*55522268
*55522269
*55522270
*55522271
*55522272
*55522273
*55522274
*55522275
*55522276
*55522277
*55522278
*55522279
*55522280
*55522281
*55522282
*55522283
*55522284
*55522285
*55522286
*55522287
*55522288
*55522289
*55522290
*55522291
*55522292
*55522293
*55522294
*55522295
*55522296
*55522297
*55522298
*55522299
*555222100
*555222101
*555222102
*555222103
*555222104
*555222105
*555222106
*555222107
*555222108
*555222109
*555222110
*555222111
*555222112
*555222113
*555222114
*555222115
*555222116
*555222117
*555222118
*555222119
*555222120
*555222121
*555222122
*555222123
*555222124
*555222125
*555222126
*555222127
*555222128
*555222129
*555222130
*555222131
*555222132
*555222133
*555222134
*555222135
*555222136
*555222137
*555222138
*555222139
*555222140
*555222141
*555222142
*555222143
*555222144
*555222145
*555222146
*555222147
*555222148
*555222149
*555222150
*555222151
*555222152
*555222153
*555222154
*555222155
*555222156
*555222157
*555222158
*555222159
*555222160
*555222161
*555222162
*555222163
*555222164
*555222165
*555222166
*555222167
*555222168
*555222169
*555222170
*555222171
*555222172
*555222173
*555222174
*555222175
*555222176
*555222177
*555222178
*555222179
*555222180
*555222181
*555222182
*555222183
*555222184
*555222185
*555222186
*555222187
*555222188
*555222189
*555222190
*555222191
*555222192
*555222193
*555222194
*555222195
*555222196
*555222197
*555222198
*555222199
*555222200
*555222201
*555222202
*555222203
*555222204
*555222205
*555222206
*555222207
*555222208
*555222209
*555222210
*555222211
*555222212
*555222213
*555222214
*555222215
*555222216
*555222217
*555222218
*555222219
*555222220
*555222221
*555222222
*555222223
*555222224
*555222225
*555222226
*555222227
*555222228
*555222229
*555222230
*555222231
*555222232
*555222233
*555222234
*555222235
*555222236
*555222237
*555222238
*555222239
*555222240
*555222241
*555222242
*555222243
*555222244
*555222245
*555222246
*555222247
*555222248
*555222249
*555222250
*555222251
*555222252
*555222253
*555222254
*555222255
*555222256
*555222257
*555222258
*555222259
*555222260
*555222261
*555222262
*555222263
*555222264
*555222265
*555222266
*555222267
*555222268
*555222269
*555222270
*555222271
*555222272
*555222273
*555222274
*555222275
*555222276
*555222277
*555222278
*555222279
*555222280
*555222281
*555222282
*555222283
*555222284
*555222285
*555222286
*555222287
*555222288
*555222289
*555222290
*555222291
*555222292
*555222293
*555222294
*555222295
*555222296
*555222297
*555222298
*555222299
*555222300
*555222301
*555222302
*555222303
*555222304
*555222305
*555222306
*555222307
*555222308
*555222309
*555222310
*555222311
*555222312
*555222313
*555222314
*555222315
*555222316
*555222317
*555222318
*555222319
*555222320
*555222321
*555222322
*555222323
*555222324
*555222325
*555222326
*555222327
*555222328
*555222329
*555222330
*555222331
*555222332
*555222333
*555222334
*555222335
*555222336
*555222337
*555222338
*555222339
*555222340
*555222341
*555222342
*555222343
*555222344
*555222345
*555222346
*555222347
*555222348
*555222349
*555222350
*555222351
*555222352
*555222353
*555222354
*555222355
*555222356
*555222357
*555222358
*555222359
*555222360
*555222361
*555222362
*555222363
*555222364
*555222365
*555222366
*555222367
*555222368
*555222369
*555222370
*555222371
*555222372
*555222373
*555222374
*555222375
*555222376
*555222377
*555222378
*555222
```

C and assembly language: example 3

```
void swap (int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
void main (void)
{
    int x = 2, y = 3;
    swap(&x, &y);
}
```

SP	TEMP	-2(A6)
A6	00	02
	OLD_A6	OLD_A6
	OLD_A6	OLD_A6
	RET_ADD	RET_ADD
	RET_ADD	RET_ADD
	&x	&x
	&x	&x
	&y	&y
	&y	&y
Y	00	03 (02)
X	00	02 (03)
	OLD_A6	OLD_A6
	OLD_A6	OLD_A6

```

*1     void swap (int *a, int *b)                                     SECTION      S_swap,, "code"
* Parameter a is at 8(A6)
* Parameter b is at 12(A6)
* Variable temp is at -2(A6)

*2         {
*3             int temp;
*4             temp = *a;
*5             *a = *b;
*6             *b = temp;
*7         }

* Function size = 30
*8     void main (void)                                              XREF        __main

* Variable x is at -2(A6)
* Variable y is at -4(A6)

*9         {
*10             int x = 2, y = 3;
*11             swap(&x, &y);
*12         }

* Function size = 34
* bytes of code = 64

*13

```

		_swap	XDEF	_swap
	000000	4e56ffffe	LINK	A6,#-2
			EQU	\$000004
*2				
*3				
*4				
*5	000000 + __P1	286e0008	MOVEA.L	8(A6),A4
	000004 + __P1	3d54ffffe	MOVE	(A4),-2(A6)
*6	000008 + __P1	206e000c	MOVEA.L	12(A6),A0
	00000c + __P1	3890	MOVE	(A0),(A4)
*7	000012 + __P1	38aeffffe	MOVEA.L	12(A6),A4
			MOVE	-2(A6),(A4)
	00001a	4e5e	UNLK	A6
	00001c	4e75	RTS	
* Function size = 30			XREF	__main
*8 void main (void)				
* Variable x is at -2(A6)			XDEF	__main
* Variable y is at -4(A6)				
*9 {				
*10 int x = 2, y = 3;				
	000000 + __P2	3d7c0002ffffe	MOVE	#2,-2(A6)
	000006 + __P2	3d7c0003ffffc	MOVE	#3,-4(A6)
*11 swap(&x, &y);				
	00000c + __P2	486efffc	PEA.L	-4(A6)
	000010 + __P2	486efffe	PEA.L	-2(A6)
	000014 + __P2	4eb9_____	JSR	_swap
*12 }				
	00003c	4e5e	UNLK	A6
	00003e	4e75	RTS	
* Function size = 34				
* bytes of code = 64				
*13			dgroup	data
			END	



C and assembly language: recursion

```

int factorial(int n)
{
    if (n==1)
        return (1);
    else
        return(factorial(n-1)*n);
}

void main()
{
    int y, count = 2;
    y = factorial(count);
}

```

SP,A6	OLD_A6	OLD_A6
	OLD_A6	OLD_A6
	RET	RET
	RET	RET
00	01	
00	00	
00	02	
OLD_A6	OLD_A6	
OLD_A6	OLD_A6	
RET_ADD	RET_ADD	
RET_ADD	RET_ADD	
00	02	
00	02	
OLD_A6	OLD_A6	
OLD_A6	OLD_A6	
---	---	

```

*1      int factorial(int n)
* Parameter n is at 8(A6)                                     SECTION      S_factorial,, "code"
                                                               XDEF         _factorial
                                                               LINK         A6,#0
                                                               EQU          $000004
                                                               _P1
                                                               CMPI         #1,8(A6)
                                                               BNE          L1
                                                               _factorial
                                                               MOVEQ.L     #1,D0
                                                               BRA          L2
                                                               _factorial
                                                               MOVE         8(A6),D1
                                                               MOVE.L       D1,-(A7)
                                                               SUBQ         #1,D1
                                                               MOVE         D1,-(A7)
                                                               JSR          _factorial
                                                               ADDQ.L     #2,A7
                                                               MOVE.L       (A7)+,D1
                                                               MULS         D1,D0
                                                               L1
                                                               _P1
                                                               MOVE         000010 + __P1 322e0008
                                                               MOVE         000014 + __P1 2f01
                                                               MOVE         000016 + __P1 5341
                                                               MOVE         000018 + __P1 3f01
                                                               MOVE         00001a + __P1 4eb9
                                                               MOVE         000020 + __P1 548f
                                                               MOVE         000022 + __P1 221f
                                                               MOVE         000024 + __P1 c1c1
                                                               L2
                                                               _P1
                                                               UNLK         A6
                                                               RTS
                                                               * Function size = 46
                                                               *8      void main()                               XREF        _main
                                                               * Variable y is at -2(A6)
                                                               * Variable count is at -4(A6)                  XDEF        _main
                                                               _main
                                                               _P2
                                                               LINK         A6,#-4
                                                               EQU          $000032
                                                               _P2
                                                               MOVE         #2,-4(A6)
                                                               MOVE         000006 + __P2 3d7c0006ffffc
                                                               MOVE         00000a + __P2 3f3c0006
                                                               MOVE         000010 + __P2 3d40fffe
                                                               JSR          _factorial
                                                               MOVE         #2,-(A7)
                                                               MOVE         D0,-2(A6)
                                                               UNLK         A6
                                                               RTS
                                                               * bytes of sdata = 0                           _dgroup
                                                               END          data
                                                               *13

```

