13/06/2017 afternoon

Iteration statements

- There are the iteration statements :
  - *while(condition) statement*
  - *do statement while (condition);*
  - *for (for-init-statement; condition; expression) statement*
- Each of these statements, iterate the statement(s) in until a *break* or *return* statement is found or the condition is false.
- In the do statement condition is tested after the block execution.
- In the for and while statements condition is tested before the block execution:

```
while(1) { // execute forever until break }
for(int i=0; i<10; i++) { //  iterate 10 times with i=0...9 }
```

Iteration statements

- for statements should be used for a fixed number of iterations or when the iteration variable is easy to chose:

```
for(int k=0; k<10; ++k) a[k]=0;
for(int k=0,j=0,i=1000; k<10; a[++k]=++j,--i) ;
```

- while statements are usually used when the iteration variable it is not easy to choose or has to be updated in the loop:

```
while(tracks < 1000) {
        if(find_tracks()) ++tracks;
```

Statement overview

- More used statements:
  - *declaration*
  - *{statement-list}*
  - *expression;*

  - *if (condition) statement*
  - *if (condition) statement else statement*
  - *switch (condition) statement*

  - *while(condition) statement*
  - *do statement while (expression);*
  - *for (for-init-statement; condition; expression) statement*
    - *condition:*
      - *expression*
      - *type-specifier declarator = expression*

Statement overview

- More used statements :
  - *case constant-expression : statement*
  - *default: statement*
  - *break;*
  - *continue;*
  - *return expression;*
  - *goto identifier;*
    - *identifier : statement*

Statement overview

- What the fact that a *declaration* is a *statement* means?

```
int main(){
  int a=5;
  a=a/2;
  int b=a;
  return 0;
}
```

I can create objects anywhere in my code!

- I can create object (variables) when I do need it.

Exercises

- What does this line do:

  *while (*a++=*b++);*

  - Write a function using it (*hint.* Let's use *char* …)
- Write a program using a function to calculate the factorial of an integer: N!
- Write the previous exercise without using any iteration statement.

Function overloading

- We often use the same name for a block of code performing the same task but acting differently according to the type.
- Using the same name for functions with different parameter type is called: *overloading*.
- For example how is the operator *++* able to act on different types having the same name?

  *int a=0,b[5]; int *p=b; *(p++)=a++;*
- Name must be the same, parameters and return type may change:

  float sqrt(float); double sqrt(double); int sqrt(int)
- Overloading resolution relies on a complicated set of rules, don't be surprised of the choice done by the compiler.

Pointers to function

- There are only two things you can do to a function: call it and take its address. The pointer obtained can be used to call the function.
- To declare a pointer to a function you must declare exactly the return and parameter types. To avoid ambiguitis with the * prefix in a declarator, the name and the * must be enclosed within parenthesis:
  - *int *dn (int); // declaration of a function named: dn, returning a pointer to an int.*
  - *int (*dn)(int); // declaration of a variable named: dn, storing pointer to a function returning an int.*

Pointers to function

- It is possible to apply the operator *&* to the function name or using the name with no operator:

```
int do_nothing(int a);
int (*dn) (int);
int main() {
    int a=2;
    dn=&do_nothing;
    dn=do_nothing;
    return dn(a);
}
```

- It is often convenient to define a name for a pointer to function type to avoid using the not obvious declaration syntax all the time.

```
typedef void (*SIG_TYP) (int); // from UNIX <signal.h>
typedef void (*SIG_ARG_TYP) (int);
SIG_TYP signal(int, SIG_ARG_TYP);
```

Pointers to function

- Very handy to implement menus using array of pointers:

```
typedef void (*PF) ();// declare PF as a name for a pointer to
    function, returning void and having no formal arguments, type
PF edit_ops[]={&cut,&paste,&copy,&search}; // edit operations
PF file_ops[]={&open,&append,&close,&write}; // file operations
PF *button2=edit_ops;
PF *button3=file_ops;
button2[1]();
edit_ops[1]();
```

Functions: Call by Value

- When a function is “called”, a memory block for every argument is created and every argument objects is initialized to its actual value.
- We pass to the function a copy of the argument’s value.

Example

- https://owncloud.ba.infn.it/public.php?service=files&t=8701bd0be133452351a80f2352a13247

```
#include "my_first_f.h"

typedef int Fscaf;

int do_nothing(int a){return a++;}

int main(){
  int a,c;
  a=c=2;
  Fscaf d;
  d=a;
  a+=c;
  {
    int d=12;
    c+=d;
  }
  c=do_nothing(a);
  return (my_first_f(a,c)+d);
}
```

Test_function.cpp

Include function declarations

```
int my_first_f(int a, int b) {
  return a+b;
}
```

my_first_f.h

Compilation & Link

Scope with local variables: d != d in the main, c == c in the main

Execute

1. ***g++ -o test_function test_function.cpp***
2. ***./test_function***
3. ***echo $?***
4. ***gdb test_function***

Check result

Functions: Call by reference

- Passing by value is efficient and simple if we want to pass small objects.
- Besides, what if we did want a function to modify its arguments?
- We can use pointers, of course, but what about having the possibility to declare a new name for an object and pass this new name to the function?

Reference

- **Reference**
  - A reference is an alias for an object. It is an alternative name for an object.
  - The notation: *T&* means: *reference to T*
    - *int i=2: int &ref=i: int a=ref; ref=2;*
  - A reference must point to an object. Initialization is mandatory:
    - *int i=0; int & ref=i; //ok*
    - *int & ref: // Error. Not initialized reference*
    - *extern int &ref2; // ok, compiler trust the fact that is initialized // somewhere else.*
  - There are no operations permitted on a reference. Every operation acts on the object:
    - ref++; // Adds 1 to the object referred to

Functions: Call by reference

- Using a reference as a function formal argument you can modify the actual object in a function.
- *Caveat*. To keep program readable:
  - It is best to avoid function that modify their argument.
  - Avoid function that modify several arguments!
  - Return values whenever is possible.
  - Always use names whit a strong hint on argument modification.

```
int incr1(int a) { return a+1; }
void incr2(int& a) { ++a; }
int x = 7;
x = incr1(x);          // obvious
incr2(x);     // obscure
```

- Prefer usage of const reference arguments.

Functions: Call by reference

- A rule of thumb:
  - Use pass by value to pass very small objects.
  - Use pass by const reference to pass large objects that you don't need to modify.
  - Return a result rather than modifying an object throught a reference argument.
  - Use pass by reference only when you have to.

```
void f(int a, int& r, const int& cr) { ++a; ++r; ++cr; }
void g(int a, int& r, const int& cr) { ++a; ++r; int x = cr; ++x; }

int main(){
   int x = 0;
   int y = 0;
   int z = 0;
   g(x,y,z);
   g(1,2,3);
   g(1,y,3);
   return 0;}
```

Correct?

Correct?

Error: reference argument r needs a variable to refer to

Correct: since cr is const we can pass "a temporary"

Example

https://owncloud.ba.infn.it/public.php?service=files&t=9e0b05675d27106e7d4185e1a6d5cfac

```
void changeit(int & a){
  a++;
}
void nochangeit(int a){
  a++;
}

int main(){
  int prova;
  prova = 500;
  nochangeit(prova);
  changeit(prova);
  return prova;
}
```

Test_reference.cpp

Execute

Compilation & Link

1. g++ -o test_reference test_reference.cpp
2. ./test_reference
3. echo $?
4. gdb test_reference

Check result

User defined type: *struct*

- Structures.
  - An array is an aggregate of objects of the same type. A structure is an aggregate of object of different types. The statement: *struct*, introduce the definition of a struct:

```
struct studente {
 char * name;
 int anno_dott;
 bool buoni_e_cattivi;
 bool corso_cpp;
};
```

  - Once defined you can use it as a base type in declarations or definitions:

```
studente f_cafagna;
```

User defined type: *struct*

- It is possible to initialize it using the same syntax used for an array:

  studente f_cafagna={"Francesco Cafagna",1,TRUE,TRUE};

- How can I individually access to a structure member?
  - You can use the operator:. *(period)*

    std::cout << f_cafagna.name << std::endl;
- But if has the same support of a base type, there are also pointers and references?
- Yes, of course.
  - If you are dealing with pointers to a reference then you have to use the *structure pointer dereference* operator: *->*

    *void Print_student_name(studente *s){*

    std::cout << s->name << std::endl;

    }

Watch out the ';'

User defined type: *struct*

- It has the same support of a base type, so it is possible to assign them.
- You can use them as function parameters or return types.
- Compiler support a default object creation, copy and assignment operations, but no other operations are defined.

```
struct Date{
    int day;
    int month;
    int year;
};

int main() {
    Date today={26,9,2016}, today1;
    Date today2=today, today3;
    today1=today;
    today3=today1+today2;
    return 0;
}
```

Default object creation

Used as base type!!!!

Copy

Correct?

Assignment

User defined type: *struct*

- A name can be used just after the declaration (see pag. 78. Grammatica di base: Declaration), so we can write:

```
struct test {
 test* prova;
 test* riprova;
};
```

  - Why this is not legal?

```
struct test {
 test prova;
 test riprova;
};
```

  - … and this one?

```
struct prova2;
struct test{
 prova2 wrong;
 test *right;
};
```

Is it a definition or a declaration?

User defined type: *union*

- Unions.
  - These are *struct*s with members sharing the same memory address.
  - Union size is the biggest member one:

```
union test {
  char t;
  int s;
};
test prova;
prova.t='y';
std::cout << prova.s << std::endl;
```

It has an *int* size

What will be the output?

Templates

- C++ support generic programming, that is it is possible to use a type that is parameter of a definition:

```
template<class T> struct MemoryChunk{
   typedef T data_type;
  T generic_variable;
  T* pointer_to_a_generic_variable;
};
template<class C> C do_nothing(C a){ return a++;}
```

- Template functions can be overloaded:

```
template<class T> T sqrt(T);
template<class T> complex<T> sqrt(complex<T>);
```

Templates

- How can I instantiate them, that is how can I specialize them?
  - Type can be specified enclosing it between < and > …
  - … or letting the compiler choose the type in case no ambiguity is present:

```
MemoryChunk <unsigned int> my_data; // explicit specialization
int a;
int b=do_nothing(a); // compiler will choose do_nothing<int>(a)
MemoryChunk c=do_nothing(my_data);
// compiler will choose do_nothing<MemoryChunk>(a)
```

- Parameters can be used in a template:

```
template <class S, int n> struct Pere{
   S v[n];
};
Pere<int,5> Williams;
```

- Parameters can be: constants, addresses of objects or external functions, pointers to not overloaded members.

Templates: example

- https://owncloud.ba.infn.it/public.php?service=files&t=882bee5ec8c77f11f240ca020a223e4f

test_templates2.cpp

```
#include "my_first_f.h"

typedef int Fscaf;

template<class C> C do_nothing(C a){return a++;}

int main(){
  int a,c;
  a=c=2;
  Fscaf d;
  d=a;
  a+=c;
  {
    int d=12;
    c+=d;
  }
  c=do_nothing(a);
  float g=0.;
  g=do_nothing(g);
  return (my_first_f(a,c)+d);
}
```

my_first_f.h

```
int my_first_f(int a, int b) {
  return a+b;
}
```

Function template definition

Specialization to int

Specialization to float

Redirecting standard output

1. ***g++ -S –fverbose-asm –g -o test_templates2.s test_templates2.cpp***
2. ***as –alhnd test_templates2.s > test_templates.list***
3. ***nano test_templates.list***

Grammatica di base

test_funtion.list

```
1                               .file   "test_templates2.cpp"
 2                # GNU C++ version 3.4.4 (cygming special, gdc 0.12, using dmd 0.125) (i686-pc-cygwin)
 3                #       compiled by GNU C version 3.4.4 (cygming special, gdc 0.12, using dmd 0.125).
 4                # GGC heuristics: --param ggc-min-expand=100 --param ggc-min-heapsize=131072
 5                # options passed:  -D__CYGWIN32__ -D__CYGWIN__ -Dunix -D__unix__ -D__unix
 6                # -idirafter -idirafter -mtune=pentiumpro -auxbase-strip -g -fverbose-asm
 7                # options enabled:  -feliminate-unused-debug-types -fpeephole
 8                # -ffunction-cse -fkeep-static-consts -freg-struct-return -fgcse-lm
 9                # -fgcse-sm -fgcse-las -fsched-interblock -fsched-spec
10                # -fsched-stalled-insns -fsched-stalled-insns-dep -fbranch-count-reg
11                # -fcommon -fverbose-asm -fargument-alias -fzero-initialized-in-bss
12                # -fident -fmath-errno -ftrapping-math -m80387 -mhard-float
13                # -mno-soft-float -malign-double -mieee-fp -mfp-ret-in-387
14                # -mstack-arg-probe -maccumulate-outgoing-args -mno-red-zone
15                # -mtune=pentiumpro -march=i386
16
19                              .text
20              Ltext0:
45                              .align 2
50              .globl __Z10my_first_fii
52              __Z10my_first_fii:
54              Lbext1:
 1:my_first_f.h  **** int my_first_f(int a, int b) {
56              LM1:
57 0000 55                                   pushl   %ebp    #
58 0001 89E5                                 movl    %esp, %ebp      #,
59              LBB2:
 2:my_first_f.h  ****   return a+b;
61              LM2:
62 0003 8B450C                               movl    12(%ebp), %eax  # b, b
63 0006 034508                               addl    8(%ebp), %eax   # a,
                tmp59
64              LBE2:
 3:my_first_f.h  **** }
66              LM3:
67 0009 5D                                   popl    %ebp    #
68 000a C3                                   ret
69              Lscope0:
74 000b 90                                   .align 2
76              .globl _main
78              _main:
80              Ltext2:
 1:test_templates2.cpp **** #include "my_first_f.h"
 2:test_templates2.cpp ****
 3:test_templates2.cpp **** typedef int Fscaf;
 4:test_templates2.cpp ****
 5:test_templates2.cpp **** template<class C> C do_nothing(C a){return a++;}
 6:test_templates2.cpp ****
 7:test_templates2.cpp **** int main(){
82              LM4:
83 000c 55                                   pushl   %ebp    #
84 000d 89E5                                 movl    %esp, %ebp      #,
85 000f 83EC28                               subl    $40, %esp       #,
86 0012 83E4F0                               andl    $-16, %esp      #,
87 0015 B8000000                             movl    $0, %eax        #,
                tmp59
87      00
88 001a 83C00F                               addl    $15, %eax       #,
                tmp60
89 001d 83C00F                               addl    $15, %eax       #,
                tmp61
90 0020 C1E804                               shrl    $4, %eax        #,
                tmp62
91 0023 C1E004                               sall    $4, %eax        #,
                tmp64
92 0026 8945E8                               movl    %eax, -24(%ebp)         #
                tmp64,
93 0029 8B45E8                               movl    -24(%ebp), %eax         #,
94 002c E8000000                             call    __alloca
94      00
96              LM5:
97 0031 E8000000                             call    ___main         #
97      00
98              LBB3:
99              LBB4:
 8:test_templates2.cpp ****   int a,c;
 9:test_templates2.cpp ****   a=c=2;
101             LM6:
102 0036 C745F802                            movl    $2, -8(%ebp)    #, c
102     000000
103 003d C745FC02                            movl    $2, -4(%ebp)    #, a
103     000000
10:test_templates2.cpp ****   Fscaf d;
11:test_templates2.cpp ****   d=a;
105             LM7:
106 0044 8B45FC                              movl    -4(%ebp), %eax  # a, a
107 0047 8945F4                              movl    %eax, -12(%ebp)         # a, d
12:test_templates2.cpp ****   a+=c;
109             LM8:
110 004a 8B55F8                              movl    -8(%ebp), %edx  # c, c
111 004d 8D45FC                              leal    -4(%ebp), %eax  #,
                tmp84
112 0050 0110                                addl    %edx, (%eax)    # c, a
113             LBB5:
13:test_templates2.cpp ****   {
14:test_templates2.cpp ****     int d=12;
115             LM9:
116 0052 C745F00C                            movl    $12, -16(%ebp)  #, d
116     000000
15:test_templates2.cpp ****   c+=d;
118             LM10:
119 0059 8B55F0                              movl    -16(%ebp), %edx         # d, d
120 005c 8D45F8                              leal    -8(%ebp), %eax  #,
                tmp86
121 005f 0110                                addl    %edx, (%eax)    # d, c
122             LBE5:
16:test_templates2.cpp ****   }
17:test_templates2.cpp ****   c=do_nothing(a);
124             LM11:
125 0061 8B45FC                              movl    -4(%ebp), %eax  # a, a
126 0064 890424                              movl    %eax, (%esp)    # a,
127 0067 E8000000                            call    __Z10do_nothingIiET_S0_         #
127     00
128 006c 8945F8                              movl    %eax, -8(%ebp)  #
                tmp75, c
18:test_templates2.cpp ****   float g=0.;
130             LM12:
131 006f B8000000                            movl    $0x00000000, %eax       #,
                tmp76
131     00
132 0074 8945EC                              movl    %eax, -20(%ebp)         #
                tmp76, g
19:test_templates2.cpp ****   g=do_nothing(g);
134             LM13:
135 0077 8B45EC                              movl    -20(%ebp), %eax         # g, g
136 007a 890424                              movl    %eax, (%esp)    # g,
137 007d E8000000                            call    __Z10do_nothingIfET_S0_         #
137     00
138 0082 D95DEC                              fstps   -20(%ebp)       # g
20:test_templates2.cpp ****   return (my_first_f(a,c)+d);
140             LM14:
141 0085 8B45F8                              movl    -8(%ebp), %eax  # c, c
142 0088 89442404                            movl    %eax, 4(%esp)   # c,
143 008c 8B45FC                              movl    -4(%ebp), %eax  # a, a
144 008f 890424                              movl    %eax, (%esp)    # a,
145 0092 E869FFFF                            call    __Z10my_first_fii       #
145     FF
146 0097 0345F4                              addl    -12(%ebp), %eax         # d,
                tmp79
147             LBE4:
148             LBE3:
21:test_templates2.cpp **** }
150             LM15:
151 009a C9                                  leave
152 009b C3                                  ret
162             Lscope1:
164                                          .section        .text$_Z10do_nothingIiET_S0_,"x"
165                                          .linkonce discard
166                                          .align 2
169             .globl __Z10do_nothingIiET_S0_
171             __Z10do_nothingIiET_S0_:
173             LM16:
174 0000 55                                   pushl   %ebp    #
175 0001 89E5                                movl    %esp, %ebp      #,
176             LBB6:
178             LM17:
179 0003 8B4508                              movl    8(%ebp), %eax   # a, a
180 0006 FF4508                              incl    8(%ebp)         # a
181             LBE6:
182 0009 5D                                  popl    %ebp    #
183 000a C3                                  ret
184             Lscope2:
186 000b 90                                  .section        .text$_Z10do_nothingIfET_S0_,"x"
187                                          .linkonce discard
188                                          .align 2
191             .globl __Z10do_nothingIfET_S0_
193             __Z10do_nothingIfET_S0_:
195             LM18:
196 0000 55                                  pushl   %ebp    #
197 0001 89E5                                movl    %esp, %ebp      #,
198 0003 83EC04                              subl    $4, %esp        #,
199             LBB7:
201             LM19:
202 0006 8D4508                              leal    8(%ebp), %eax   #,
                tmp60
203 0009 D900                                flds    (%eax)  # a
204 000b D9E8                                fld1
205 000d D9C9                                fxch    %st(1)  #
206 000f D955FC                              fsts    -4(%ebp)        #
207 0012 8B55FC                              movl    -4(%ebp), %edx  #, a
208 0015 DEC1                                faddp   %st, %st(1)     #,
209 0017 D918                                fstps   (%eax)  # a
210 0019 8955FC                              movl    %edx, -4(%ebp)  # a,
211 001c D945FC                              flds    -4(%ebp)        #
212             LBE7:
213 001f C9                                  leave
214 0020 C3                                  ret
215             Lscope3:
217 0021 909090                              .text
219             Letext:
```


INFN Istituto Nazionale di Fisica Nucleare Sezione di Bari

```
...
  8:test_templates2.cpp ****   int a,c;
  9:test_templates2.cpp ****   a=c=2;
101             LM6:
102 0036 C745F802       movl    $2, -8(%ebp)    #, c
102      000000
103 003d C745FC02       movl    $2, -4(%ebp)    #, a
103      000000
 10:test_templates2.cpp ****   Fscaf d;
 11:test_templates2.cpp ****   d=a;
105             LM7:
106 0044 8B45FC         movl    -4(%ebp), %eax # a, a
107 0047 8945F4         movl    %eax, -12(%ebp)     # a, d
...
 17:test_templates2.cpp ****   c=do_nothing(a);
124             LM11:
125 0061 8B45FC         movl    -4(%ebp), %eax # a, a
126 0064 890424         movl    %eax, (%esp)    # a,
127 0067 E8000000         call  __Z10do_nothingIiET_S0_     #
127      00
128 006c 8945F8         movl    %eax, -8(%ebp) # tmp75, c
 18:test_templates2.cpp ****   float g=0.;
130             LM12:
131 006f B8000000       movl    $0x00000000, %eax  #, tmp76
131      00
132 0074 8945EC         movl    %eax, -20(%ebp)     # tmp76, g
 19:test_templates2.cpp ****   g=do_nothing(g);
134             LM13:
135 0077 8B45EC         movl    -20(%ebp), %eax     # g, g
136 007a 890424         movl    %eax, (%esp)    # g,
137 007d E8000000         call  __Z10do_nothingIfET_S0_     #
137      00
138 0082 D95DEC         fstps   -20(%ebp)   # g
...
```

```
162             Lscope1:
164                 .section    .text$_Z10do_nothingIiET_S0_,"x"
165                 .linkonce discard
166                 .align 2
169             .globl __Z10do_nothingIiET_S0_
171             __Z10do_nothingIiET_S0_:
173             LM16:
174 0000 55              pushl   %ebp   #
175 0001 89E5            movl    %esp, %ebp    #,
176             LBB6:
178             LM17:
179 0003 8B4508          movl    8(%ebp), %eax    # a, a
180 0006 FF4508          incl 8(%ebp)    # a
181             LBE6:
182 0009 5D              popl    %ebp   #
183 000a C3              ret
```

```
184             Lscope2:
186 000b 90              .section
        .text$_Z10do_nothingIfET_S0_,"x"
187                 .linkonce discard
188                 .align 2
191             .globl __Z10do_nothingIfET_S0_
193             __Z10do_nothingIfET_S0_:
195             LM18:
196 0000 55              pushl   %ebp   #
197 0001 89E5            movl    %esp, %ebp    #,
198 0003 83EC04          subl    $4, %esp   #,
199             LBB7:
201             LM19:
202 0006 8D4508          leal 8(%ebp), %eax    #, tmp60
203 0009 D900            flds(%eax)      # a
204 000b D9E8            fld1
205 000d D9C9            fxch    %st(1)  #
206 000f D955FC          fsts-4(%ebp)   #
207 0012 8B55FC          movl    -4(%ebp), %edx   #, a
208 0015 DEC1            faddp   %st, %st(1)    #,
209 0017 D918            fstps   (%eax)      # a
210 0019 8955FC          movl    %edx, -4(%ebp)   # a,
211 001c D945FC          flds-4(%ebp)   #
212             LBE7:
213 001f C9            leave
214 0020 C3              ret
215             Lscope3:
217 0021 909090          .text
219             Letext:
```

test_funtion.list

```
 69                 Lscope0:
 73 000b 90                 .align 2
 76                 .globl __Z10do_nothingi
 78                 __Z10do_nothingi:
 80                 Ltext2:
  5:test_function.cpp **** int do_nothing(int a){return a++;}
 82                 LM4:
 83 000c 55                 pushl   %ebp    #
 84 000d 89E5               movl    %esp, %ebp      #,
 85                 LBB3:
 87                 LM5:
 88 000f 8B4508             movl    8(%ebp), %eax   # a, a
 89 0012 FF4508             incl 8(%ebp)       # a
 90                 LBE3:
 91 0015 5D                 popl%ebp    #
 92 0016 C3                 ret
...
 17:test_function.cpp ****   c=do_nothing(a);
144                 LM13:
145 006d 8B45FC           movl    -4(%ebp), %eax # a, a
146 0070 890424           movl    %eax, (%esp)    # a,
147 0073 E894FFFF           call __Z10do_nothingi      #
147      FF
148 0078 8945F8           movl    %eax, -8(%ebp) # tmp75, c
 ...
```

test_funtion.list

```
162                 Lscope1:
164                     .section    .text$_Z10do_nothingIiET_S0_,"x"
165                     .linkonce discard
166                     .align 2
169                 .globl __Z10do_nothingIiET_S0_
171                 __Z10do_nothingIiET_S0_:
173                 LM16:
174 0000 55                 pushl   %ebp   #
175 0001 89E5               movl    %esp, %ebp    #,
176                 LBB6:
178                 LM17:
179 0003 8B4508             movl    8(%ebp), %eax    # a, a
180 0006 FF4508             incl 8(%ebp)    # a
181                 LBE6:
182 0009 5D                 popl    %ebp   #
183 000a C3                 ret
184                 Lscope2:
186 000b 90                 .section
    .text$_Z10do_nothingIfET_S0_,"x"
187                     .linkonce discard
188                     .align 2
191                 .globl __Z10do_nothingIfET_S0_
193                 __Z10do_nothingIfET_S0_:
195                 LM18:
196 0000 55                 pushl   %ebp   #
197 0001 89E5               movl    %esp, %ebp    #,
198 0003 83EC04             subl    $4, %esp   #,
199                 LBB7:
201                 LM19:
202 0006 8D4508             leal8(%ebp), %eax    #, tmp60
203 0009 D900               flds(%eax)      # a
204 000b D9E8               fld1
205 000d D9C9               fxch    %st(1)  #
206 000f D955FC             fsts-4(%ebp)   #
207 0012 8B55FC             movl    -4(%ebp), %edx   #, a
208 0015 DEC1               faddp   %st, %st(1)    #,
209 0017 D918               fstps   (%eax)      # a
210 0019 8955FC             movl    %edx, -4(%ebp)   # a,
211 001c D945FC             flds-4(%ebp)   #
212                 LBE7:
213 001f C9             leave
214 0020 C3                 ret
215                 Lscope3:
217 0021 909090             .text
219                 Letext:
```