# II. Program structure and variables

Vasili Slapik
vasili_slapik@epam.com

17 февраля 2014 г.

## Variables

- identifiers are case sensitive
- identifiers must start with a letter
- should not start with an underscore
- should consist of 31 or fewer characters to ensure portability
- should not be a keyword

# C language keywords

| | | |
|---|---|---|
| auto | if | unsigned |
| break | inline (C99) | void |
| case | int | volatile |
| char | long | while |
| const | register | _Alignas (C11) |
| continue | restrict (C99) | _Alignof (C11) |
| default | return | _Atomic (C11) |
| do | short | _Bool (C99) |
| double | signed | _Complex (C99) |
| else | sizeof | _Generic (C11) |
| enum | static | _Imaginary (C99) |
| extern | struct | _Noreturn (C11) |
| float | switch | _Static_assert (C11) |
| for | typedef | _Thread_local (C11) |
| goto | union | |

## Constants and literals

The constants refer to fixed values that the program may not alter during its execution. These fixed values are also called literals.

- numeric
  - decimal: **1234**
  - octal: **01234**
  - hexidecimal: **0x1234**
  - floating-point: **123.456e-67**
  - hexidecimal floating-point: **0x1.999999999999ap-4** (0.1)
- character: **'A'**
- string: **"A"**, **"Hello world"**

## Constants types

- integer constants

| | | |
|---|---|---|
| no prefix | 44 | **int** |
| U | 55U | **unsigned int** |
| L | 66L | **long int** |
| LL | 77LL | **long long int** |

- float-point constants

| | | |
|---|---|---|
| no prefix | 4.0 | **double** |
| F | 6.6F | **float** |
| L | 75e3L | **long double** |

## Constants types

```c
#include <stdio.h>

int main(void)
{
    printf("value %zd\n", sizeof(5));
    printf("value %zd\n", sizeof(5L));
    printf("value %zd\n", sizeof(5LL));
    printf("value %zd\n", sizeof(5.0));
    printf("value %zd\n", sizeof(5.0F));
    printf("value %zd\n", sizeof(5.0L));
    printf("value %zd\n", sizeof('x'));

    return 0;
}
```

## Defining constants

- using **#define** macro

```c
#define FALSE       0
#define BUFFER_SIZE 20
```

- using **const** keyword

```c
int const a = 1234;
const int a = 4321;
```

## Backslash escapes

| | |
|---|---|
| \\ | Literal backslash |
| \" | Double quote |
| \' | Single quote |
| \n | Newline (line feed) |
| \r | Carriage return |
| \b | Backspace |
| \t | Horizontal tab |
| \f | Form feed |
| \a | Alert (bell) |
| \v | Vertical tab |
| \? | Question mark (used to escape trigraphs) |
| \nnn | Character with octal value nnn |
| \xhh | Character with hexadecimal value hh |

## C data types

- **char**: at least 8 bit, `sizeof(char) == 1`, CHAR_BIT macro
- **short**: at least 16 bit, greater or equlal to `sizeof(char)`
- **int**: at least 16 bit, greater or equlal to `sizeof(short)`
- **long**: at least 32 bit, greater or equlal to `sizeof(int)`
- **long long**: C99, at least 64 bit, greater or equlal to `sizeof(long)`
- **bool**: C99, at least one bit, without <stdbool.h> - **_Bool**
- **float**: single precision floating-point type, at least 6 decimal digits
- **double**: single precision floating-point type, at least 10 decimal digits
- **long double**: extended precision floating-point type if available, otherwise it is the same as double
- **signed/unsigned** modifiers
- **complex** modifier, C99, without <complex.h> - **_Complex**

## Common traps

```c
#include <stdio.h>

int main(void)
{
    for (unsigned int i = 9; i >= 0; i--)
    {
        printf("%d", i);
    }

    return 0;
}
```

## Common traps

```c
#include <stdio.h>

int ping(unsigned char a, unsigned char b,
         unsigned char c, unsigned char d)
{
    // do all stuff

    return 0;
}

int main()
{
    ping(192, 168, 121, 221);
    ping(192, 168, 121, 121);
    ping(192, 168, 121, 021);

    return 0;
}
```

## Common traps

```c
/* [-m32] */
#include <stdio.h>

int main(void)
{
    long int a = -1;
    unsigned int b = 1;

    if (a > b)
        puts("a");
    else
        puts("b");

    return 0;
}
```

## 64 data models

| Model | int | long | long long | pointers | Sample operation systems |
|---|---|---|---|---|---|
| LLP64 | 32 | 32 | 64 | 64 | MS Windows (x86-64 and IA-64) |
| LP64 | 32 | 64 | 64 | 64 | Most Unix and Unix-like systems, Solaris, Linux, BSD, OS X, z/OS |
| ILP64 | 64 | 64 | 64 | 64 | HAL Computer Systems port of Solaris to SPARC64 |

## C additional data types

- **size_t/ssize_t, ptrdiff_t**
- **int8_t, int16_t, int32_t, int64_t**
- **uint8_t, uint16_t, uint32_t, uint64_t**
- ... and many others from the <stdint.h>

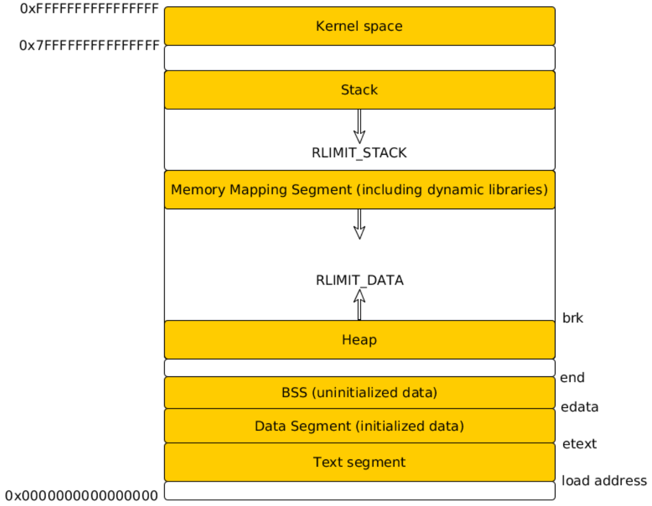
## Storage classes

- automatic
- static
- allocated
- register
- extern
- thread (since C11), variables declard with `_Thread_local` keyword

## Storage classes

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

int a;              // data segment (bss)
int b = 1;          // data segment
static int c;       // data segment (bss)
static int d = 2; // data segment
extern int errno; // extern data

void func(void)
{
    register int j;         // register or stack (automatic)
}

int main(void)
{
    static char i[1024 * 1024 * 128L] = {0}; // data segment
    short int k;            // stack (automatic)
    auto long int m;        // stack (automatic)
    void *ptr = malloc(16); // 16 bytes in the heap, ptr in stack

    return 0;
}
```

# Memory layout



## Memory layout

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

#define SIZE (1024 * 1024 * 16)

int main()
{

    char a[1024 * 1024 * 16];
    int i;

    for (i = 0; i < SIZE; i++)
    {
        a[i] = 0xcc;
    }

    printf("I'm here !!!\n");

    return 0;
}
```

## Memory layout

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>

extern char etext, edata, end;

int main(int argc, char *argv[])
{

    printf("First address past\n");
    printf("    program text (etext):      %20p\n", &etext);
    printf("    initialized data (edata): %20p\n", &edata);
    printf("    uninitialized data (end): %20p\n", &end);
    printf("Program break (brk):          %20p\n", sbrk(0));
    printf("Address of argc:              %20p\n", &argc);
    printf("Address of errno:             %20p\n", &errno);
    printf("Address of printf:            %20p\n", printf);
    printf("Address of malloced memory:   %20p\n", malloc(16));

    exit(EXIT_SUCCESS);
}
```