# Programmētāju skola
## 3. līmeņa grupa

**dati**

# Informācijas mērvienības

| | |
|---|---|
| 0 vai 1 | 1 bits |
| 8 biti, piem. $1001\ 0011_2$ | 1 baits |
| $100\ 0000\ 0000_2 = 2^{10}$ baits | 1K = 1 kilobaits = 1024 baits |
| $2^{10}$K baits = $2^{20}$ baits | 1M = 1 megabaits = 1 048 576 baits |
| $2^{30}$ baits | 1G = 1 gigabaits |
| $2^{40}$ baits | 1T = 1 terabaits |
| $2^{23}$ baits | $2^{23} = 2^3 \cdot 2^{20} = 8$M |

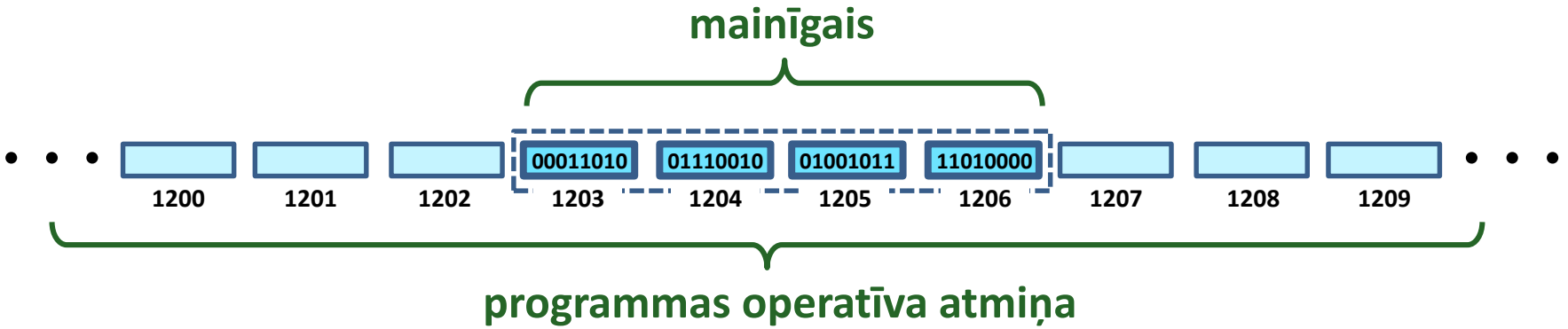
# Mainīgie un konstantes

**Konstante** – identifikators, kas vērtība nevar būt izmainīta programmas darba gaitā.

`326, 12.23` – literāļi

`PI, SIZE_OF_ARRAY` – nosauktas konstantes

**Mainīgais** – identifikators, kas vērtība izmainās programmas darba gaitā.



# Literāļi

**Veselo skaitļu konstantes:**

decimālā forma `5  12  100  -765`

oktālā forma `05  014  0144  -01375  0976`

heksadecimālā forma `0x5  0xC  0x64  -0x2FD`

binārā forma `0b101  0b11001010  0b1100101011110101`

**Reālo skaitļu konstantes:**

tradicionālā forma `5.  .0  1.2  -0.001  -765.654`

eksponenciālā forma `1e2  12.34e1  12.34e-1`

**Simbolu konstantes:**

drukājamā forma `'5'  'A'  '.'  '?'`

nedrukājamā forma `'\n'  '\t'  '\x41'='A'  '\101'='A'`

**Simbolu rindas konstantes:**

šauras rindas (*narrow strings*) `"5"  "Hello"  "Hello\n"  ""  "\"Quoted text\""`

jēlas rindas (*raw strings*) `R"("Quoted text")"  R"(D:\Student\ddb002\)"`

**Loģiskas konstantes:** `true  false`

**Ciparu grupēšana:** `0b1001'1001'0100  178'954'234`

# Nosauktas konstantes

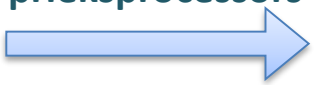
Literāļu lietošana pasliktina koda lasāmību un sarežģi tā izmaiņas. Laba prakse ir izmantot nosauktās konstantes literāļu vietā.

## C stila konstantes (*makro aizstāšana*)

```
#define NAME value
```

```
#define MAX_SIZE 1024
...
if (size > MAX_SIZE) ...
```

priekšprocessors



```
...
if (size > 1024) ...
```

## C++ stila konstantes

```
const type NAME = value;
```

```
const int MAX_SIZE = 1024;
...
if (size > MAX_SIZE) ...
```

# Mainīgie

```
modifier type name;
```

| | | |
|---|---|---|
| **Tipi:** | `int` | veselo skaitļu tips ar zīmi |
| | `float` | vienkārtīgas precizitātes reālo skaitļu tips (7 cipari) |
| | `double` | divkāršas precizitātes reālo skaitļu tips (15 cipari) |
| | `char` | simbolu tips (vienbaitu veselo skaitļu tips ar zīmi) |
| | `bool` | loģiskais tips (true/false) |
| **Modifikatori:** | `long/short` | palielina/samazina izmēru atmiņā un vērtības diapazonu |
| | `signed/unsigned` | nobīda vērtības diapazonu negatīvā/pozitīvā pusē |
| | `const` | vērtība nevar būt izmainīta (konstante) |
| **Vārdi:** | `name_of_variable` | Snake Case style |
| | `NAME_OF_VARIABLE` | Screaming Snake Case style |
| | `NameOfVariable` | Upper Camel Case style jeb Pascal style |
| | `nameOfVariable` | Lower Camel Case style |

**Piemēram:**

```
int i;
float weight, height;
unsigned char letter;
bool found;
unsigned long long int hugeInteger;
```

# Mainīgo inicializācija

Uzreiz pēc izveides mainīgais saturēs kādu nejaušu vērtību, kas atradās atmiņas vietā, kurā mainīgais tika izveidots. Tādu vērtību sauc pār nenoteikto vērtību (*undefined value*). Darbs ar nenoteiktām mainīgo vērtībām var izraisīt neparedzamu programmas uzvedību (*undefined behavior*). Lai noteiktu mainīgā sākumvērtību var noradīt to deklarācijas laikā (*initialization*).

```
modifier type name = value;
```

```
modifier type name ( value );
```

```
modifier type name { value };
```

**Piemēram :**

```
int i = 0;
float weight(1.5), height(-0.5);
unsigned char letter{ 'A' };
bool found{ true };
unsigned long long int hugeInteger{ 9'223'372'036'854'775'807 };
```

Ja mainīgais tiek inicializēts deklarācijas laikā, tipa nosaukumu var aizstāt ar atslēgvārdu `auto`, kas liek kompilatoram noteikt mainīgā tipu, pamatojoties uz inicializācijas vērtību.

**Piemēram :**

```
auto size = 0;
auto string{ "Initial text" };
auto hugeInteger{ 9'223'372'036'854'775'807 };
```

# Bāziskie tipi

| Type Name | Bytes | Other Names | Range of Values |
|---|---|---|---|
| **char** | 1 | signed char | –128 … 127 |
| **unsigned char** | 1 | | 0 … 255 |
| **short int** | 2 | short<br>signed short int | –32,768 … 32,767 |
| **unsigned short int** | 2 | unsigned short | 0 … 65,535 |
| **int** | 4 | signed | –2,147,483,648 … 2,147,483,647 |
| **unsigned int** | 4 | unsigned | 0 … 4,294,967,295 |
| long int | 4 | long<br>signed long int | same as int |
| unsigned long int | 4 | unsigned long | same as unsigned int |
| **long long int** | 8 | long long | –9,223,372,036,854,775,808 …<br>9,223,372,036,854,775,807 |
| **unsigned long long int** | 8 | unsigned long long | 0 … 18,446,744,073,709,551,615 |
| **float** | 4 | | $\pm 3.4 \cdot 10^{\pm 38}$ (7 digits) |
| **double** | 8 | | $\pm 1.7 \cdot 10^{\pm 308}$ (15 digits) |
| long double | 8 | | same as double |
| **bool** | 1 | | false or true |

# Aritmētiskas operācijas

**Saskaitīšana:** `a + b`

```
5 + 2 → 7
```

**Atņemšana:** `a - b`

```
5 - 2 → 3
```

**Reizināšana:** `a * b`

```
5 * 2 → 10
```

**Dalīšana:** `a / b`

```
5 / 2 → 2
5. / 2. → 2.5
5 / 2. → 2.5
```

**Dalīšanas atlikums:** `a % b`

```
5 % 2 → 1
```

# Samazināšana un palielināšana

| Palielināšana: a ++ | Samazināšana: a -- |
|---|---|
| a ++ → a = a + 1 | a -- → a = a - 1 |

Palielināšanas un samazināšanas operācijām ir divas formas:

**postfiksālā** (a++ un a--) un **prefiksālā** (++a un --a)

Postfiksālā formā mainīgais izmainās **pēc izteiksmes aprēķina**, prefiksālā formā – **pirms aprēķina**.

T.i. postfiksālā operācijas rezultāts ir **neizmainīta vērtība**, bet prefiksālā – **izmainīta vērtība**.

```
a = 5; b = a--;          a=4, b=5
a = 5; if (a++ > 5)…     a=6, false
a = 5; c[a++] = 1;       a=6, c[5]=1
```

```
a = 5; b = --a;          a=4, b=4
a = 5; if (++a > 5)…     a=6, true
a = 5; c[++a] = 1;       a=6, c[6]=1
```

Postfiksālā operācijas rezultāts ir **vērtība**, prefiksālā – **reference** jeb **mainīgais**.

```
a = 5; ++++a;      a=7
a = 5; --a = 0;    a=0
```

```
a = 5; a++++;      needs l-value
a = 5; a-- = 0;    needs l-value
```

# Piešķiršanas operācijas

**Piešķiršana:** `a = b`

```
a = 5;   a = b;   a = b + 2;   a = a + 2;
```

piešķiršanas operācijas rezultāts ir piešķirtais **mainīgais**:

```
a = b = c = 5;
if ((a = b) > 10)...
```

**Aritmētiskas piešķiršanas:**

```
a += b → a = a + b
a -= b → a = a - b
a *= b → a = a * b
a /= b → a = a / b
a %= b → a = a % b
```

```
a = 5; a += 2;     a=7
a = 7; a /= 3;     a=2
```

visu piešķiršanas operāciju rezultāti ir **references**:

```
(a = 5) ++;                 a=6
a = 5; (a += 5) *= 2;       a=20
```

# Salīdzināšanas operācijas

**Salīdzināšanas operācijas :**

```
a > b
a >= b
a < b
a <= b
a == b
a != b
```

```
5 > 10 → false = 0

5 < 10 → true = 1
```

```
int a=1,b=-1,c=-2;  cnt = (a < 0) + (b < 0) + (c < 0); → cnt = 2

int a=1;  v = 10 * (a == 0) + 20 * (a == 1); → v = 20

int a=1,b=2;  max = a * (a > b) + b * (a <= b); → max = 2
```

# Loģiskas operācijas

## operācija UN: a && b

| a | b | a && b |
|---|---|---|
| =0 | =0 | 0 |
| =0 | ≠0 | 0 |
| ≠0 | =0 | 0 |
| ≠0 | ≠0 | 1 |

```
int a=1; v = a > -10 && a < 10; → v = true = 1

v = 5 && 0; → false = 0
```

## operācija VAI: a || b

| a | b | a \|\| b |
|---|---|---|
| =0 | =0 | 0 |
| =0 | ≠0 | 1 |
| ≠0 | =0 | 1 |
| ≠0 | ≠0 | 1 |

```
int a=1; v = a < -10 || a > 10; → v = false = 0

v = 5 || 0; → true = 1
```

## operācija NE: ! a

| a | !a |
|---|---|
| =0 | 1 |
| ≠0 | 0 |

```
int a=1; v = !(a < -10 || a > 10); → v = true = 1

v = !5; → false = 0
```

# Nosacīta jeb ternārā operācija

```
a ? b : c   →   if ( a ) b; else c;
```

```
a = a < 0 ? 0 : a;
```

```
max = a > b ? a : b;
```

```
if ( ( a > b ? a : b ) == 10 ) ...
```

```
max = a > b && a > c ? a : ( b > c ? b : c );
```

```
(a > b ? a : b)++;
```

```
printf("a is %s\n", a < 0 ? "negative" : "pozitive");
```

# Masīvi

```
modifier type name [ size ] { values };
```

```
int array[5];
float w[4] { 0.5, 1.5, -1.5 };                  // ceturtais elements ir 0
bool states[] { true, true, false, true };      // masīva izmērs ir 4
char hi[80] { 'H', 'e', 'l', 'l', 'o' };        // pārēji 75 simboli ir 0
int buf[100] {};                                // visi elementi ir 0
char string[] { "Hello" };                      // masīva izmērs 6
```

```
name [ index ]
```

```
array[3] = 1;
w[2] = w[1] + 2;
if (hi[0] == 'H') hi[0] = 'h';
```

# Gadījumskaitļi

C bibliotēkā `stdlib` ir gadījumskaitļu ģenerators, ko var izmantot pseidogadījumskaitļu ģenerēšanai.

Darbam ar gadījumskaitļiem ir jāpielieto sekojošas funkcijas:

**`rand()`** ģenerēt gadījumskaitļi diapazonā no `0` to `RAND_MAX` (`32767`)

```
int v = rand();
int v = rand() % 100;
int v = rand() % 101 - 50;
```

**`srand(v)`** inicializēt pseidogadījumskaitļu ģeneratoru ar skaitli `v`

```
srand( 12 );
srand( time(NULL) );
```

```
srand( time(NULL) ); // vienreiz programmas sākumā
int a[100];
for (int i = 0; i < 100; i++) a[i] = rand() % 21 - 10;
```