**Примитивне структуре података су основне структуре и програмски језици поседују уграђену подршку за рад са њима, па се зато у литератури, за ове типове података, користи назив уграђени подаци (Built-in Data).**

***Integer*, целобројни тип података,**

***Floating point*, децимални број,**

***Boolean* (*true* – тачно, *false* – нетачно), логички тип података,**

***Character & Strings*, знаковни тип података.**

**Подаци типа *integer* (*int*) су целобројне вредности.**

**Целобројне константе, променљиве, изрази и функције.**

**Опсег целобројне вредности је различит и може се мењати применом квалификатора *long*, *short* и *unsigned*.**

***long* повећава опсег вредности целобројних променљивих.**

***short испред int смањује* опсег целобројних променљивих.**

***unsigned* испред *int* декларише променљиву тако да може меморисати само позитивне вредности.**

| Назив типа податка | Опис | Величина | Интервал |
|---|---|---|---|
| **int** | Integer – цео број | минимум 16 бита | **- 32 767 ÷ 32 767** |
| **short (short int)** | Short integer – кратки цео број | минимум 16 бита | **- 32 767 ÷ 32 767** |
| **unsigned int** | Цео позитивни број | | **0 ÷ 65 535** |
| **long (long int)** | Long integer – durački ceo broj | минимум 32 бита | **-2 147 483 647 ÷ 2 147 483 647** |
| **unsigned long** | Дугачки цео позитиван број | минимум 32 бита | **0 ÷ 4 294 967 295** |
| **long long int** | Продужени дугачки цели број | минимум 64 бита | **–9 223 372 036 854 775 807 ÷ 9 223 372 036 854 775 807** |
| **unsigned long long int** | **Продужени дугачки цели позитиван број** | **минимум 64 бита** | **0 ÷ 18 446 744 073 709 551 615** |

**Карактеристика овог типа примитивне структуре података, у програмском језику *C*, је да резултат рачунарских операција, са промењљивима овог типа је увек целобројан.**

***Пример.***

**Израчунати резултат основних математичких операција за дате вредности целобројних промењљивих *x* и *y*.**

***x* = 35; *y* = 17**

***1.* Сабирање: *rez* = *x* + *y*; *rez* = 35 + 17 = 52**

***2.* Одузимање: *rez* = *x* - *y*; *rez* = 35 - 17 = 18;**

***rez* = *y* – *x*; *rez* = 17 - 35 = - 18;**

***3.* Множење: *rez* = *x* * *y*; *rez* = 35 * 17 = 595;**

***4.* Дељење: *rez* = *x* / *y*; *rez* = 35 / 17 = 2;**

***rez* = *y* / *x*;*rez* = 17 / 35 = 0;**

***Float*** **тип податка, за своје вредности узима подскупове реалних бројева.**

**Овај тип података обухвата један коначан подскуп рационалних бројева ограничене величине и тачности.**

***float*** **тип података који се најближе приближава појму реалног броја.**

**Све меморијске локације коначне дужине, па стога и бројни подаци који се у њих смештају морају бити коначне дужине и тако по природи ствари отпадају ирационални бројеви.**

| Назів типа податка | Опис | Величина минимално у бајтовима | Интервал |
|---|---|---|---|
| **float** | Реални број | 4 | **$\pm 3.4 \cdot 10^{\pm 38}$, са тачношћу од приближно 6 децимала** |
| **double** | Дугачки реални број | 8 | **$\pm 1.7 \cdot 10^{\pm 308}$, са тачношћу од приближно 16 децимала** |
| **long double** | **Продужени дугачки реални број** | **8** | **$\pm 1.7 \cdot 10^{\pm 308}$, са тачношћу од приближно 19 децимала** |

**Карактеристика *float* типа, у програмском језику *C*, је та да је резултат рачунарских операција увек реалан број.**

**Резулат се даје у децималном запису са приближном тачношћу броја децимала за изабрани тип (за *float* то је 6 децимала: 15.234000 или 15.000000).**

***Пример.***

**Израчунати резултат основних математичких операција за дате вредности реалних променљивих *x* и *y*.**

***x* = 35.; *y* = 17.**

***1.* Сабирање: *rez* = *x* + *y*; *rez* = 35. + 17. = 52.000000**

***2.* Одузимање: *rez* = *x* - *y*; *rez* = 35. – 17. = 18.000000;**
***rez* = *y* – *x*; *rez* = 17. – 35. = - 18.000000;**

***3.* Множење: *rez* = *x* * *y*; *rez* = 35. * 17. = 595.000000;**

***4.* Дељење: *rez* = *x* / *y*; *rez* = 35 / 17 = 2.058824;**
***rez* = *y* / *x*;*rez* = 17 / 35 = *0.485714;***

**Како заокружити на жељени број децимала.**

***Пример.***

**Израчунати резултат основних математичких операција за дате вредности реалних променљивих *x* и *y*. Резултат заокружити на 2 децимале.**

***x* = 35.; *y* = 17.**

***1.* Сабирање: *rez* = *x* + *y*; *rez* = 35. + 17. = 52.00**

***2.* Одузимање: *rez* = *x* - *y*; *rez* = 35. – 17. = 18.00;**
***rez* = *y* – *x*; *rez* = 17. – 35. = - 18.00;**

***3.* Множење: *rez* = *x* * *y*; *rez* = 35. * 17. = 595.00;**

***4.* Дељење: *rez* = *x* / *y*; *rez* = 35 / 17 = 2.06;** **2.058824;**
***rez* = *y* / *x*;*rez* = 17 / 35 = 0.48;** **0.485714;**

**Програмски језик аутоматски заокружује последњу децималу по правилима математичког заокруживања, које гласи.**

**Ако је прва цифра коју одбацујемо мања или једнака 4, тада цифре лево од ње не дирамо:**

**3,48943… ≈ 3**
**7,098923… ≈ 7**
**54.6724594156… ≈ 54.67**

**Ако је прва цифра коју одбацујемо већа од 5, или је једнака 5, а након ње нису све нуле, тада прву цифру лево од ње, то јест, последњу од оних које задржавамо, увећавамо за један:**

**2,631884… ≈ 3**
**14,92034… ≈ 15**
**81.2935285961… ≈ 81.294**

**Граничан случај је када је прва (и једина) цифра коју одбацујемо петица, након које следе све нуле. Тада је правило да, ако је цифра лево од одбачене петице парна, онда се она не дира, а ако је непарна, тада се увећава за један, како би постала парна:**

**3,5≈4**
**2,3245≈2,324**
**6,42375≈6,4238**

***Пример.***

**Оператор % - за одређивање остатка дељења два ЦЕЛА броја.**

***int x* = 35; *y* = 17;**

***rez* = *x* % *y*; *rez* = 35 % 17 = 2;**

***rez* = *y* % *x*; *rez* = 17 % 35 = 17;**

***float x* = 35.; *y* = 17.;**

***rez* = *x* % *y*; *rez* = 35. % 17. = ⇨ *Error – грешка синтаксе***

***rez* = *y* % *x*; *rez* = 17. % 35. = ⇨ *Error – грешка синтаксе***

**Логички типови података постоје као основни типови података у свим програмским језицима.**

**LOGICAL или BOOLEAN.**

**Обухватају само две вредности *true* (тачно) и *false* (нетачно).**

**Основне логичке операције *not*, *and*, *or*, *xor* и *eqv*.**

<table>
<tr><th>Логичка операција</th><th colspan="2">Опис</th><th>Оператор</th></tr>
<tr><td rowspan="4">not</td><td colspan="2">Негира BOOLEAN вредност. На пример „not true” враћа „false”, а „not false“ враћа „true”.</td><td rowspan="4">!</td></tr>
<tr><td>A</td><td>not A<br>(!A)</td></tr>
<tr><td>true</td><td>false</td></tr>
<tr><td>false</td><td>true</td></tr>
</table>

<table>
<tr><th>Логичка операција</th><th colspan="3">Опис</th><th>Оператор</th></tr>
<tr><td rowspan="6">and</td><td colspan="3">Упоређује две BOOLEAN вредности и враћа „true“ уколико су само обе вредности „true“. У супротном враћа „false“.</td><td rowspan="6">&&<br>или<br>&</td></tr>
<tr><td>A</td><td>B</td><td>A and B<br>(A && B)</td></tr>
<tr><td>true</td><td>true</td><td>true</td></tr>
<tr><td>true</td><td>false</td><td>false</td></tr>
<tr><td>false</td><td>true</td><td>false</td></tr>
<tr><td>false</td><td>false</td><td>false</td></tr>
</table>

<table>
<tr><th>Логичка операција</th><th colspan="3">Опис</th><th>Оператор</th></tr>
<tr><td rowspan="6">or</td><td colspan="3">Упоређује две BOOLEAN вредности и враћа „true“ уколико су упоређене вредности различите. Уколико су обе вредности исте враћа се „false“.</td><td rowspan="6">||</td></tr>
<tr><td>A</td><td>B</td><td>A or B<br>(A || B)</td></tr>
<tr><td>true</td><td>true</td><td>true</td></tr>
<tr><td>true</td><td>false</td><td>true</td></tr>
<tr><td>false</td><td>true</td><td>true</td></tr>
<tr><td>false</td><td>false</td><td>false</td></tr>
</table>

<table>
<tr><th>Логичка операција</th><th colspan="3">Опис</th><th>Оператор</th></tr>
<tr><td rowspan="6">xor</td><td colspan="3">Упоређује две BOOLEAN вредности и враћа „true“ уколико су упоређене вредности различите. Уколико су обе вредности исте враћа се „false“.</td><td rowspan="6">^</td></tr>
<tr><td>A</td><td>B</td><td>A xor B<br>(A ^ B)</td></tr>
<tr><td>0</td><td>0</td><td>false</td></tr>
<tr><td>0</td><td>1</td><td>true</td></tr>
<tr><td>1</td><td>0</td><td>true</td></tr>
<tr><td>1</td><td>1</td><td>false</td></tr>
</table>

***BOOLEAN, Логички* – тип податка**

<table>
<tr><th>Логичка операција</th><th colspan="3">Опис</th><th>Оператор</th></tr>
<tr><td rowspan="6">eqv</td><td colspan="3">Еквиваленција упоређује две BOOLEAN вредности и враћа „true“ уколико су обе вредности исте. Запазите разлику у односу на xor оператор.</td><td rowspan="6">==</td></tr>
<tr><td>A</td><td>B</td><td>A eqv B<br>(A == B)</td></tr>
<tr><td>0</td><td>0</td><td>true</td></tr>
<tr><td>0</td><td>1</td><td>false</td></tr>
<tr><td>1</td><td>0</td><td>false</td></tr>
<tr><td>1</td><td>1</td><td>true</td></tr>
</table>

***BOOLEAN, Логички* – тип податка**

**Изрази у којима се примењују логичке променљиве и константе називају се логички изрази.**

**Ако се логичке константе означавају са *true* и *false* онда подразумевамо да сви изрази морају бити сачињени стриктно од логичких величина.**

**Израз** ***true* – ТАЧНО**

$$Z = (x > 0)\ and\ (\ (y == 1)\ or\ (y < 0)\ )????$$

***false* – НЕТАЧНО**

**Подразумева се и да су неисправни мешовити изрази у којима се користе величине *true* и *false* помешане са нумеричким константама и аритметичким операцијама, односно свим другим типовима података.**

***Пример.***

| A | B | C |
|---|---|---|
| 15.23 | 15.23 | 35.21 |

| Израз | Опис | Резултат |
|---|---|---|
| A==B | Да ли је A еквивалентно B | true - тачно |
| A==C | Да ли је A еквивалентно C | false - нетачно |
| A!=C | Да ли је A различито од C | true - тачно |
| (A>=B) | Да ли је A веће или једнако B | true - тачно |
| (A<C) | Да ли је A мање од C | true - тачно |

***Пример.***

| A | B | C |
|---|---|---|
| **15.23** | **15.23** | **35.21** |

| | | | |
|---|---|---|---|
| **(A==B) && (A>C)** | **Да ли је A еквивалентно B И да ли је A веће од C** | **A==B** | **A>C** |
| | | ***true*** | ***false*** |
| | | **&&** | |
| | | ***false*** | |
| **(A==B) \|\| (A>C)** | **Да ли је A еквивалентно B ИЛИ да ли је A веће од C** | **A==B** | **A>C** |
| | | ***true*** | ***false*** |
| | | **\|\|** | |
| | | ***true*** | |

**Корисник рачунара комуницира са рачунаром преко улазних и излазних уређаја и ту је битно да се јављују подаци у форми која је читљива за човека.**

**То значи да се комуницирање обавља помоћу знакова из одређене азбуке која редовно обухвата уазбучена велика и мала слова, цифре, специјалне и контролне знаке.**

**( ) ! # C % & * _ - + = : “ ; ‘ < > ? , / . [ ] { } ^**

**Ови типови података се представљју између апострофа или наводника, да би се разликовали од симболичких назива променљивих.**

**О ЧЕМУ ТРЕБА ВОДИТИ РАЧУНА: A односно 'A'**

**A представља симболички назив променљиве, док 'A' представља бинарно кодирано прво слово абецеде.**

**Бинарно кодирање знакова.**

**Најпознатији метод је амерички стандард ASCII (American Standard Code for Information Interchange).**

**У оквиру ASCII стандара, корисницима је омогућено да уносе и знакове који су карактеристични за изабрану тастатуру.**

**У програмским језицима скуп знакова се назива стринг (*string*).**

**У неким програмским језицима довољно је доделити вредност променљиви, овог типа, тако што ће се иза знака додељивања (=), између апострофа или знака навода, уписати скуп знакова, на пример *a* = “Petar 1234”.**

**У програмском језику C је то мало другачије.**

**Да би се изградила секвенца карактера “ Petar 1234”, потребно је конструисати ову секвенцу помоћу основног знаковног типа податка, који се у C – у назива, *char*.**

***char*** **је основни знаковни тип податка.**

**Сваки карактер, у рачунару, се смешта у једном бајту меморије.**

$$\text{1 bajt = 8 bit} \Rightarrow 2^8 = \text{256 карактера}$$

**Ако испред декларације char стоји резервисана реч *signed*, тада се одређује интервал кодних вредности [-128,127]; ако је испред char наведено *unsigned*, тада се одређује кодни интервал [0,255].**

**Вредности промењљиве типа char, могу бити дужине једног и само једног карактера.**

**Операције које је могуће изводити са овим типом података су:**

**операције поређења:**

**< (мање од)**

**<= (мање или једнако од)**

**> (веће од)**

**>= (веће или једнако од)**

**== (једнако – еквивалентно)**

**!= (различито – нееквивалентно)**

**Операције које је могуће изводити са овим типом података су:**

**операције конверзије :**

**char у int, при чему ова операција враћа кодни број тог знака по ASCII стандарду**

## Пример

**Дате су променљиве типа char:**
е типа *char*:

$$x = "A", y = "4", m = "a", n = "7", r = "A".$$

**попунити табелу одговорима *true* или *false*.**е типа *char*:

| Релација | Одговор | Објашњење |
|---|---|---|
| $x == m$ | *false* | Кодни број (*x*) = 65; Кодни број (m) = 97. Па је исказ *x* == *m* нетачан (*false*). |
| $x < m$ | *true* | Кодни број (*x*) = 65; Кодни број (*m*) = 97. Па је исказ *x* < *m* тачан (*true*). |
| $y > r$ | *false* | Кодни број (*y*) = 52; Кодни број (*r*) = 65. Па је исказ *y* > *r* нетачан (*false*). |
| $x == r$ | *true* | Кодни број (*x*) = 65; Кодни број (*r*) = 65. Па је исказ *x* == *r* нетачан (*false*). |
| $(x>m)\&\&(y<r)$ | *false* && *true* = *false* | Кодни број (*x*) = 65; Кодни број (*m*) = 97. Па је исказ *x* > *m* нетачан (*false*).<br>Кодни број (*y*) = 52; Кодни број (*r*) = 65. Па је исказ *y* <= *r* тачан (*true*). |
| $(n<y) \,\|\| (r<=x)$ | *false* \|\| *true* = *true* | Кодни број (*n*) = 55; Кодни број (*y*) = 52. Па је исказ *n* < *y* нетачан (*false*).<br>Кодни број (*r*) = 65; Кодни број (*x*) = 65 Па је исказ *r* <= *x* тачан (*true*). |

**Стринг променљива може да има произвољан број карактера, укључујући и стринг без карактера.**

***Стринг*** **променљива може да има произвољан број карактера, укључујући и *стринг* без карактера – *празан стринг*.**

**Број карактера у *стрингу* се назива дужина *стринга*.**

**Типичне операције над стринговима су:**

- **поређење на једнакост (equality comparison);**
- **селекција карактера у стрингу (character selection);**
- **селекција подстринга (substring selection);**
- **дужина (length);**
- **лексикографско поређење (lexicographic comparison);**
- **надовезивање (concatenation);**
- **конверзија стринга у нумеричку вредност и обратно.**

***Стринг*–ови су увек знаковни тип података и граде се помоћу типа података *char*.**

**Карактеру у стрингу може да се приступи или као елементу низа или помоћу поинтера на *char*.**

**Сваки *стринг* се завршава карактером '\0'.**

***Стринг* *s* = "IVAN" може да се запамти у меморију као низ од 5 карактера *s*[0] = 'I', *s*[1] = 'V', *s*[2] = 'A', *s*[3] = 'N', *s*[4] = '\0'**

***Стрингови* су низови карактера, па се могу иницијализовати на месту декларације.**

**Пример. Декларисање стрингова:**

**Иницијализација константном вредношћу:**

*char s*[] = "STRUKTURE "

**Иницијализација "знак по знак":**

*char s*[] = {"S", "T", "R", "U", "K", "T", "U", "R", "E", "\0"}

***Стринг s*, има 10 елемената:**

| indeks | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| вредност | "S" | "T" | "R" | "U" | "K" | "T" | "U" | "R" | "E" | "\0" |

$i = 0 \Rightarrow s[0] =$ "S" ", по реду 1 елемент

$i = 4 \Rightarrow s[3] =$ "K", по реду 5 елемент

$i = 7 \Rightarrow s[7] =$ "R", по реду 8 елемент