Програмски језик C

Маја Савић

Садржај

Пример 2:

Написати програм који рачуна збир два броја која корисник уноси.

Добро, овај задатак ћемо почети постепено да радимо као и прошли, а то значи да прво морамо да унесемо локацију одакле ћемо да узимамо функције и да јасно дефинишемо циљеве програма. Те локације се називају предпроцесорске директиве. У овом слиучају користимо исте библиотеке, stdio.h и conio.h

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

Разлика између овог примера и прошлог је што сад имамо улазне параметре и тиме отварамо ново поље које се зове …

КОНСТАНТЕ , ПРОМЕНЉИВЕ И ОПЕРАТОРИ И ТИПОВИ ПОДАТАКА

Наш циљ је да нам програм сабере два броја и да нам испише на екрану њихову суму.
Добро, то је барам лако:

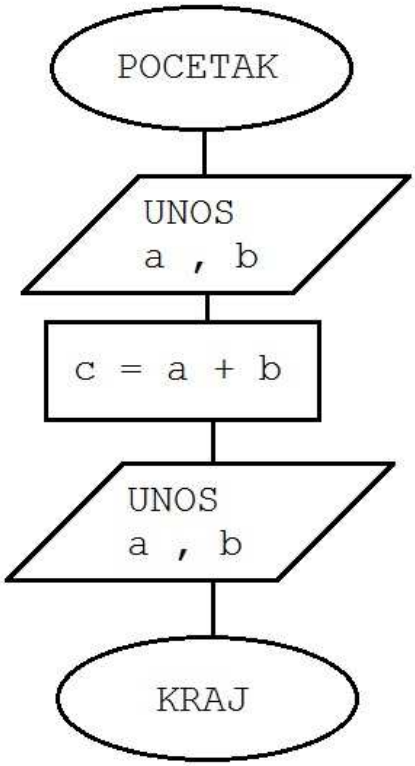
```
а = 3
б = 2
а + б = 5
```

крај….

Када би то било тако лако…

Други корак у овом задатку је формирање алгоритма (под овим се подразумева да уценици већ знају да их пишу и мисле алгоритамски, у случају да то није тако, ово може помоћи: http://goo.gl/pqK0I).

Наш алгоритам је приказан на слици поред.
Улазни параметри су бројеви а и б који се уносе преко тастатуре , а на излазу очекујемо број ц који настаје збиром а и б.



Па да наставимо са кодовањем:

Да би могли да унесемо неки број у наш програм ми морамо негде да га сместимо па тек онда користимо, а пре тога ,да дефинишемо место где га смештамо,

Зато куцамо…

```
#include<stdio.h>
#include<conio.h>
void main(void)
```

```
{
	int a , b , c ;
```

Новина је кодна линија : `int a , b , c ;`

Да би разумели шта она представља морамо пре тога да пређемо област :

Типови података.

Подаци су предмет обраде у програму , они су материјали за прављење резултата. Сваки податак је одређен особинама које чине тај (такав) тип података. Тип података је одређен:

1. Скупом могућих вредности које може да узме податак
2. Скупом операција које могу да се изврше над подацима

Типови података се могу поделити на основне и изведене (структуирани подаци). Основни типови података у C-у су целобројни и рационални типови :

Целобројни типови су:

Назив типа	Меморијски простор	Опсег
int	2 bajta	-2147483648 до 2147483647
long (int)	4 bajta	-2147483648 до 2147483647
short (int)	2 bajta	-32768 до 32767
unsigned (int)	2 bajta	0 до 65 535
char	1 bajt	0 до 255

Рационални типови :

Назив типа	Опсег	Опсег
float	4 bajta	+/- 3.4e +/- 38
Double	8 bajta	+/- 1.7e +/- 308

Од свих ових типова које смо набројали, ми ћемо за сада само анализирати тип int ,а касније ћемо и остале типове.

Тип int је основни целобројни тип података и представља целобројан позитиван или негативан број . Како заузима 2 бајта = 2*8 битова то је 216 -1 = 32 767 горња граница опсега тог бројног типа података ,а одузима се 1 јер улази и број 0.

Када смо декларисали ПРОМЕНЉИВЕ које ћемо користити у даљем раду, време је да обезбедимо улазе. Настављамо са кодовнајем и то: …

```
	printf("Unesite prvi broj a :");
	scanf("%d",&a);
```

Овим отварамо врата двема новим функцијама од којих смо једну још пре само спомињали.

ФУНКЦИЈЕ ЗА ИСПИС И УНОС ПОДАТАКА

За унос података у Ц-у користимо функцију сцанф, која има синтаксу :

Где је:
SCANF – Назив функције коју позивамо
Атрибут – Слово типа у ком цемо учитати податак
Аргумент – Уписујемо адресу где желимо да са податак смести

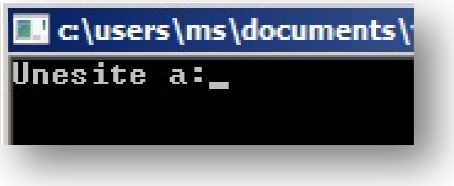
За сваки тип података постоји одговарајуци тип конверзије који се могу представити на следеци знаци:

Тип конверзије	**Тип податка**
%d	Int
%i	Int
%l	Long
%u	Unsigned
%f	Float (број у стандардном облику)
%e	Float (број у експоненцијалном облику)
%lf	Double
%s	String
%c	Char

Истим типом којим смо декларисали податке (у овом случају ћемо користити само једну и зато су сви подаци истог типа) тражимо тип конверзије и убацујемо у синтаксу наше функције са знацима навода која изгледа овако:

scanf(“%d”….

То је што се тиче **Аргумент**. Позива се касније детаљније прећи) и сачувамо. У нашем случају ми броја а и б , зато дотерујемо



атрибута. Следечи члан синтаксе је адресним оператором **&** (који ћемо именом податка где желимо да га желимо да унесемо вредности два насу синтаксу за унос првог броја…

scanf(“%d”,&a);

Покретањем програма на екрану добијамо поруку и наш програм чека унос.

Да направимо резиме:

```
scanf ("ATRIBUT" , ARGUMENT);
```

- **Атрибут** у функцији сканф представља у ком облику желимо да упишемо податак.
- **Аргумент** у функцији представља где желимо да упишемои наш податак.,

За испис података користимо функцију printf коју смо на првом примеру објаснили, а сад ћемо је само поновити.
У овом случају, printf функција нема улогу да нам испише неке податке већ да кориснику испише поруку шта програм захтева од њега. Кад смо научили и разумели да унесемо податке за обраду, урадимо

исто то и за други број:

Што изгледа овако:

```
printf("Unesite prvi broj a :");
scanf("%d",&a);
printf("\nUnesite drugi broj b:");
scnaf("%d",&b);
```

У другом испису поруке на екрану убацили смо један од контролних знакова (секвенци) који представљају беле знакове или алармира:

Ознака у коду	**...представља ово**
\n	Прелазак у следећи ред
\t	Хоризонтални табулар
\r	Повратак на почетак реда
\b	Повратак sa једно место у назад
\a	аларм

Тако да наш знак \n сигналира да следећу поруку исписује у један ред испод.
Овим завршавамо унос података, сад иде обрада. У нашем примеру обрада је сабирање зато поново куцамо …

```
c = a + b;
```

…И отварамо једно веома обивно поље ..

КОНСТАНТЕ , ПРОМЕЉИВЕ И ОПЕРАТОРИ

Константе

Прво ћемо кренути од константи :

- Константе представљају величине , које не мењају вредност током извршења програма.

У програмском језику Ц имамо 5 врсти константи:

1. **Целеобројне константе**
2. **Константе представљане у покреном зарезу**
3. **Знаковне константе**
4. **Симболичке контанте**
5. **Литерали**

1. Целобројне константе су :

а) Децималне константе [10 , 152 , -1 , 124987]
б) Окталне константе [010 , 033 , 02473]
ц) Хексадецималне константе [0XD , 0X11 , 0X4CF1]

2. Константе представљене у покретном зарезу :

Састоје се из 3 дела:
а) Било који рационални број (мантиса)
б) Слово Е или е
ц) ± целобројна вредност (експонент)

Пример:
2.23E2 ≈ 2.23 * 10 ^2 = 223
-0.41E3 ≈ -0.41 * 10 ^3 = -410

3. Знаковне константе :

Су представљене једним јединим знаком , који се пише између 2 знака апострофа. Тај знак може бити слово, знак или цифра.
Пример:
‘a’ , ‘>’ ,’4 ‘… итд

4. ЛИТЕРАЛИ
Представљају знаковне низове чија је вредност експлицитно дата. Пишу се унутар “”
Пример:
”Унесите н:”…

5. СИМБОЛИЧКЕ КОНСТАНТЕ

Константа , којој је додељно неко име, се у програмском језику Ц назива симболичка константа . Дефинису се са #define . Пишу се углавном после предпроцесорских наредби.

ПРОМЕНЉИВЕ

У програмирању , променљива је представљена именом неке меморијске локације, чију вредност је могуће мењати. Променљиве су улазни , излазни и помоћни подаци које користимо приликом рада. Тако су бројеви а , б и ц општи бројеви, чије вредности на почетку нису дефинисане али им се морају дефинисати типови да би програм знао које врсте података касније да уписе у њега.
Да појаснимо:
Желимо да податку чије је име **ПОДАТАК** дамо вредност 10 прво сто морамо да урадимо је:
1. **Дефинишемо тип (јер програм не зна да чита мисли корисника)**
2. **Именујемо га неким именом (као што се ја зовем Маја, тако се и податак зове ПОДАТАК)**
3. **Додамо му неку вреност.**

Што би изгледало овако:

```
int PODATAK;
PODATAK = 10;
```

Овим смо заправо дефинисали константу у програму јер ће његова вредност сваки пут бити иста. Зато постоји функција scanf() која нам омогућује да убацимо други број у променљиву ПОДАТАК сваки пут кад покренемо програм.

Додела вредности се извршава **ОПЕРАТОРОМ ДОДЕЛЕ** , нама познатији као знак једнако. Овим улазимо у поље оператора.

Оператори

Представљају симболе , који означавају одређену операцију и који повезују један или више операнада у израз. У зависности од броја операнада који један оператор повезује деле се не :
Унарне , бинарне и терналне.
(Асоциативност иде после)
На основу операција које врше , операторе је могуће поделити у следеће групе:

1. Аритметичке
- Негативни предзнак
+ Оператор сабирања
- Оператор одузимања
* Оператор множења
/ Оператор дељења
% Модуо броја

2. Оператори поређења
< Мање
<= Мање или једнако
> Веће
>= Веће или јендако
== Једнако
!= Различито
Сви ови оператори су бинарни

3. Логички оператори
! Негација
&& Логичко И
|| Логичко ИЛИ

4. Оператори за операције над битовима
Прескачемо

5. Оператор гранања
Једини терналнi операнд.(Радићемо после)

6. sizeof – оператор
Касније

7. Оператори за додељивање вредности

а) Елементарни операнти

Представљен је симболом “=”.
Пример : Int a = 10;

б) Сложени операнти

a = a + 2 ; a += 2;
a = a -2 ; a -= 2;
a = a * 2 ; a *= 2;
a = a / 2; a /= 2;

Ова табела важи само ако је вредност а декларисана на почетку и да има већ неку постојећу вредност.
Бинарни је оператор.

8. Инкрементирање и декрементирање

а) Инкрементација
int i = 2;
i++;
Сада је : i=3;
б) Декрементација
int i = 2;
i--;
Сада је : i=2;
Ови оператори важе само за променљиве целобројних типова и да поменљива има познату вредност. Оператор је унаран.

9. **COMMA** – оператор
Оператор набрајања, ништа посебно.

10. **CAST** – оператор
Користи се за експлицитно претварање једног типа у други тип променљиве. Унарни је оператор.
Пример:

```
int n;
(char)n;
```

11. Адресни и индиректни оператори
Употреба при раду са адресама и показивачима

12. Индексни оператори
Код низова

13. Оператори за приступ елементима структуре

Сад кад знамо шта су променљиве ,константе и оператори, можемо да наставимо са кодирањем. Оно што нам је остало је излаз. Треба да нам на екрану буде исписана нека вредност која није константна.
Рецимо :
То значи да нам је и излаз променљива. А како ћемо да испишемо вредност ПРОМЕНЉИВЕ на екрану? Вратићемо се на кратко на функцију printf(). Поред њене стандардне синтаксе да се у заградама под знацима навода пише текст који желимо да нам буде на екрану, постоји још један начин исписа вредности које нису константне:

Стављање типа конверзије у тексту и после затворених наводника одвојити зарезом и ставити име променљиве које желимо да нам се прикаже. Кодујемо..

```
printf("\nZbir vasa dva broja je: %d",c);
```

Заступљен је тип конверзије %d у изразу јер смо променљиву (њихов збир)декларисали типа int (погледати табелу са типовима конверзије), И после зареза стављамо место са ког желимо да очитамо вредност.

И на крају стављамо getch(), да нам се слика заустави и погледамо резултате, и овим завршавамо задатак. Тако да наш готов задатак изгледа овако:

```
#include<stdio.h>
#include<conio.h>
int main()
{
	int a ,b , c;
	printf("Unesite a :");
	scanf("%d",&a);
	printf("Unesite b :");
	scanf("%d",&b);
	c = a + b;
	printf("\nZbir vasa dva broja je: %d",c);
	getch();
	return 0;
}
```