Упражнение 2 – Преговор 2

Функции за вход и изход

Функциите за вход и изход позвлоляват комуникацията с потребителите. Чрез функцията за вход се позволява на потребителя да въвежда данни в програмата – като параметри,текст и др. Функцията за изход позволява извежданенето на данни в подходяща за четене и разбиране от потребителя. Функциите за вход и изход в програмният език С за сходни като идея, различават се по източника, където ще четат/записват данни. Най-често използваните са функциите:

- printf – за извеждане на данни;
- scanf – за четене на данни;

Тези функции използват т. нар. „стандартен вход и изход“ това означава, че програмата ще чете данни от клавиатурата (стандартен вход), ако не е пренасочен, и ще извежда данни на екрана (стандартен изход).

Двете функции използват т.нар. форматиращ низ, в който се задават правилата на форматиране на входа/изхода. Форматирането се използва, за да се преобразуват данните в последователност от символи и обратно. Форматиращият низ налага правилата за преобразуване.

Форматиращият низ може да съдържа в себе си текст и форматни спецификатори – специална комбинация от символи, които задава правилата дадени данни да бъдат изведени/прочетени по определен начин.

Най-често използваните форматните спецификатори са:

Форматен спецификатор	Употреба
%d	Четене/извеждане на целочислени числа
%u	Четене/извеждане на беззнакови целочислени числа
%o	Четене/извеждане на беззнакови целочислени числа в осмична бройна система
%x	Четене/извеждане на беззнакови целочислени числа в шестнадесетична бройна сиситема (с малки букви)
%X	Четене/извеждане на беззнакови целочислени числа в шестнадесетична бройна сиситема (с големи букви)
%f	Четене/извеждане на реални числа
%c	Четене/извеждане на единичен символ
%s	Четене/извеждане на символен низ

Употреба на функциите за вход и изход

printf – функция за извеждане на данни на стандартния изход

```
printf(<форматиращ низ>, <променлива 1>, <променлива 2>,...);
```

хон.ас. инж. Боян Петров, ***b_petrov@tu-sofia.bg***,
София, 2013

- **<форматиращ низ>** - съдържа правилата за извеждането на данните, съдържа комбинация от символи и форматни спецификатори, разгледани по-горе.
    - Пример: *"Stoynostta na X e %d"*
-
- **<променлива 1>, <променлива 2>,...** – тук се описват имената на променливите, чийто стойностти трябва да бъдат изведени на екрана.
- **специални символи**

Символ	Значение
\a	Издава звук
\n	Нов ред
\r	Връщане в началото на реда
\t	Табулация

- **Пример за printf:**

```
printf("Stoynostta na X e: %d", x);
```

scanf – функция за четене на данни от стандартния вход

```
scanf(<форматиращ низ>, <местоположение на променлива 1>, <местоположение на променлива 2>,...);
```

- **<форматиращ низ>** - съдържа правилата за извеждането на данните, съдържа комбинация от символи и форматни спецификатори, разгледани по-горе. Форматиращият низ може да съдържа символи, различни от формарните спецификатори, но те се интерпретират като разделители, които се игнорират при четене. Чрез символите могат да бъдат зададени шаблони (нар. още регулярни изрази), по които може да се четат данните като формати на телефони: +<код на държава> <код на град> <телефонен номер> +359 2 9877777
    - Пример: *"+%d %d %d"*

- **<местоположение на променлива 1>, <местоположение на променлива 2>,...** – тук се описват местата на променливите, където да бъдат записани прочетените данни. В програмният език С има оператор, който дава това местоположение на променливите. Операторът е "&", като за разлика от побитовата операция И, не използва две променливи, а само една. Операторът за по-лесно може да се чете като „на адреса на".
    - **Пример за & -** &x се чете като „на адреса на x"
    - **Забележка –** При четене на символни низове не се налага търсене на местоположение, този случай ще бъде разгледан в следващите упражнения.
- **Пример за scanf:**

```
scanf("%d %d",&a,&b);
```

Задачи за упражнение 1

1. Да се напише програма, която намира лице на правоъгълник по зададени от клавиатурата две страни и го извежда на екрана.
2. Да се напише програма, която намира обиколката на окръжност по зададен диаметър от клавиатурата и извежда резултата на екрана.

Логически оператори

Както беше споменато в предишното упражнение логическите операции са част от операциите, с които може да се работи с елементарните типове данни.

В програмният език C под „истина“ се разбира ненулева стойност, а под „лъжа“ – нула.

Видове логически операции

- **Логическо НЕ „!“ –** за разлика от побитовото НЕ където се прави преобразуване на ниво битове, логическото НЕ променя истинността на дадена променлива т.е. ако стойността е „лъжа“, то НЕ „лъжа“ е истина, и обратно.
- **Логическо И „&&” –** за да е „истина“ едно условие, трябва лявата и дясната страна на оператора да са „истина“
- **Логическо ИЛИ „||” –** за да е „истина“ едно условие, трябва да поне едно от двете стойности от ляво или дясно да е „истина“

Сравнения

Оператор	Значение
<	По-малко
>	По-голямо
>=	По-малко или равно
<=	По-голямо или равно
!=	Различно (не равно)
==	Равно

Забележка: *Да се внимава при употребата на сравняването за равенство, защото често се обърква с операторът за присвояване „=”. Резултатите от тази грешка могат да бъдат непредсказуеми.*

Примери

- (x > 5) && (x<=10)
- (x < 4) || (x >12)
- (x > 4) && (x!= 6)

Оператор if

В повечето програмни езици присъства възможност за разклоняване логиката на програмата – възможността да се изпълняват дадена част от кога само при изпълнено дадено условие, ако това условие не е изпълнено, кодът не се изпълнява.

В програмният език C тази възможност се реализира чрез оператора if. Употребата се извършва по начин, показан по-долу:

```
if(<Условие>)
{
	// Код за изпълнение, ако условието е изпълнено
}
```

Под <Условие> се разбира логическо условие, което може да бъде „истина“ или „лъжа“.

Оператор if-else

Употребата на този оператор е идентична на оператор if, но тук е предвиден код, който може да се изпълни само, когато условието не е изпълненено.

```
if(<Условие>)
{
	// Код за изпълнение, ако условието е изпълнено
}
else
{
	// Код за изпълнение, ако условието не е изпълнено
}
```

По този начин програматата има различно поведение в зависимост от истинността на поставеното условие.

Оператор switch-case

Идеята на този оператор е подбна на if, но тук има повече от две възможни стойности за тестване, като за всяка възможна стойност (случай) се изпълняват различни част от кода. Употреба:

```
switch(<променлива>)
{
	case <стойност 1>:
		// Код за изпълнение 1
	case <стойност 2>:
		// Код за изпълнение 2
		break;
	case <стойност 3>:
		// Код за изпълнение 3
		break;
	default:
		// Код за изпълнение 4
		break;
}
```

Значение на елементите:

- **<променлива>** - тук се записва името на променливата, чиято стойност ще бъде изследвана. Типът на променлива трябва да бъде запължително изброим тип (например цяло число).
- **<стойност 1>, <стойност 2>, … –** константи, които показват случаите, за които ще се изследва променливата.
- **case –** запазена дума, която показва от къде започва изпълнението на кода, който се отнася за определената стоност. **Изпълнението продължава до срещане на запазената дума**

***break,* възжможно е break да се намира в следващ случай (както е показано в примера), така ако променливата има стойност 1, то ще се изпълни „Код за изпълнение 1“ и „Код за изпълнение 2“.**

- **default –** в този случай се влиза, ако стойността на променливата не е описана в предходните случаи.

Задачи за упражнение 2

1. Да се прочете от клавиатурата едно целочислено числено число и да се изведе съобщение, ако то е по-голямо от 6.
2. Да се прочете от клавиатурата едно целочислено число и да се издене на екрана съобщение, ако остатъкът от деленението му с 8 е по-голям от 4.
3. Да се прочетат от клавиатурата 3 числа и да се изведе на екрана най-малкото от тях.
4. Да се прочете от клавиатурата цифра и да се изведе на екрана нейното име. Да се изполва оператор switch-case.
5. Да се прочете от клавиатурата едно число от 1 до 7 и да се изведе на екрана кой ден от седмицата съответвства на това число. Да се предвиди случай, когато не е въведено валидно число.

Създаване на функции

Функциите по своят смисъл представляват изнесени части от програмен код, наименован с име, като цел имат да намалят повтаряемостта на код, който би затруднил четенето и би удължил ненужно програмния код.

Функциите в програмирането са взаимствани от математиката, където фукнциите имат резултат от изпълнението си и определен брой променливи, за които се извършва изчислението.

Структура на функция в програмния език C

Всяка функция, която се реализира в C има структура като тази:

```
<тип> <име>(<списък с параметри>)
{
	//Код за изпълнение
	return <резултат от изпълнението>;
}
```

Където:

- **<тип> -** това е типът на резултата, който функцията ще върне. Типът може да е най-разнообразен – могат да се използват елементарните типове от данни в C, разгледани в предходното упражнение, както и потребителски типове. Възможно е някои функции да не се налага да връщат резултат, тогава в полето <тип> се записва “void”.
- **<име> -** това е името на функцията, чрез която ще бъде използвана в програмата.
- **<списък с параметри> -** параметри, които ще се използват във функцията, които са записани един след друг, като всеки параметър си има тип и име.

- **върнат резултат –** всяка функция, която има тип, различен от void трябва задължително да върне резултат от изпълнението си. Този резултат се използва от програмата, където извикана тази функция.

Пример за функция

Да се напише функция, която връща лицето на окръжност по зададен радиус.

```
float areaOfCircle(float radius)
{
	float result;
	result = radius*radius*3.1415926;
	return result;
}
```

Код на програмата:

```
#include <stdio.h>

float areaOfCircle(float radius)
{
	float result;
	result = radius*radius*3.1415926;
	return result;
}

int main()
{
	float radius1, radius2;
	float area1, area2;
	radius1 = 6.0;

	printf("Vavedete radius: ");
	scanf("%f",&radius2);

	area1 = areaOfCircle(radius1);
	area2 = areaOfCircle(radius2);

	printf("Parvata okrajnost s radius %f ima plosht: %f\n",radius1,area1);
	printf("Vtorata okrajnost s radius %f ima plosht: %f\n",radius2,area2);

	return 0;
}
```

Задача за упражение 3

Да се напише програма, в която се въвежда цяло число, което представлява код на геометрична фигура – 1 – квадрат, 2 – правоъгълник, 3 – правоъгълен триъгълник, 4 – окръжност. След това се въвеждат размерите на избраната геометрична фигура. Програмата трябва да изведе типа на геометричната фигура, както и нейното лице. За пресмятането на лицето да се напишат подходящи функции.