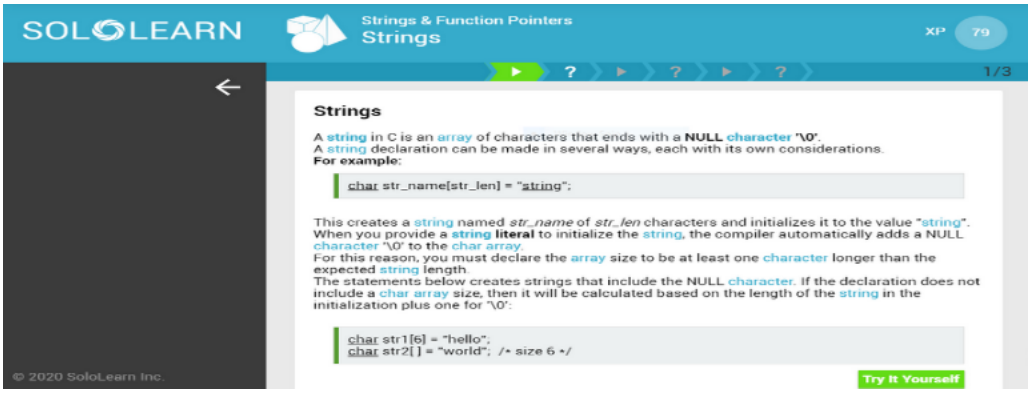
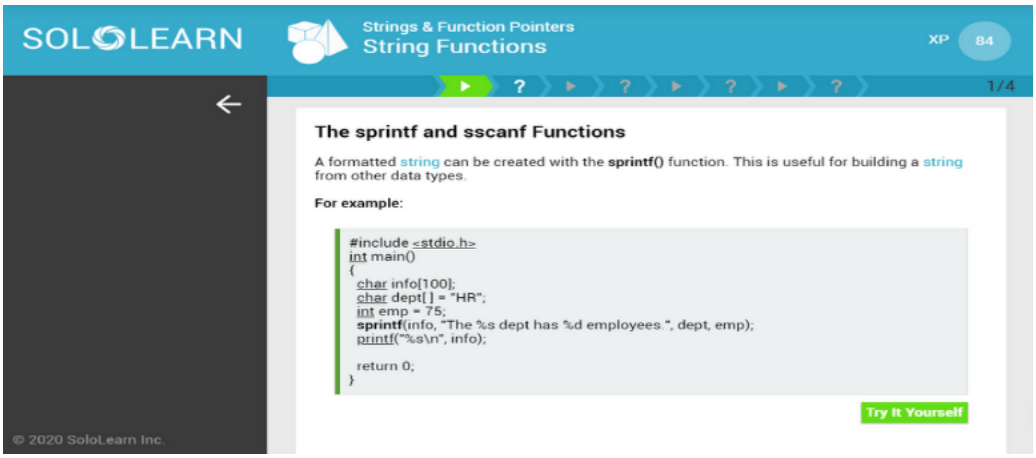
# DAILY ASSESSMENT FORMAT

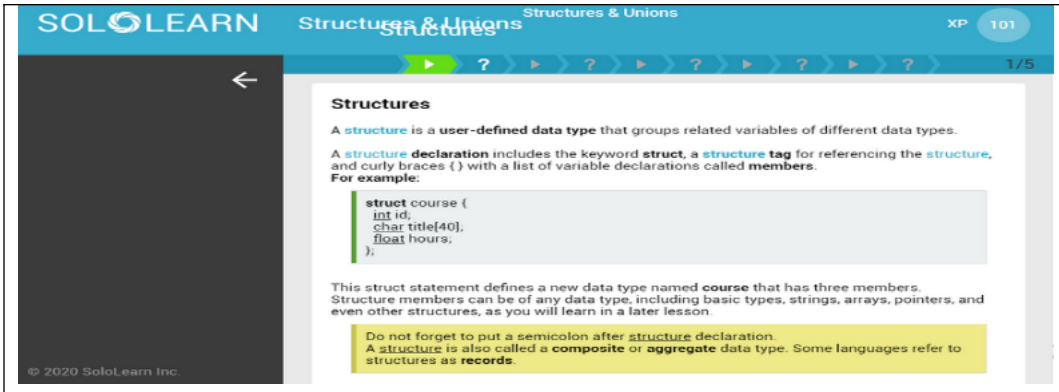
| | | | |
|---|---|---|---|
| Date: | 19-06-2020 | Name: | Rajeshwari Gadagi |
| Course: | Solo Learn | USN: | 4AL17EC076 |
| Topic: | C Programming | Semester & Section: | 6 SEM & 'B' SEC |
| Github Repository: | Rajeshwari-gadagi | | |

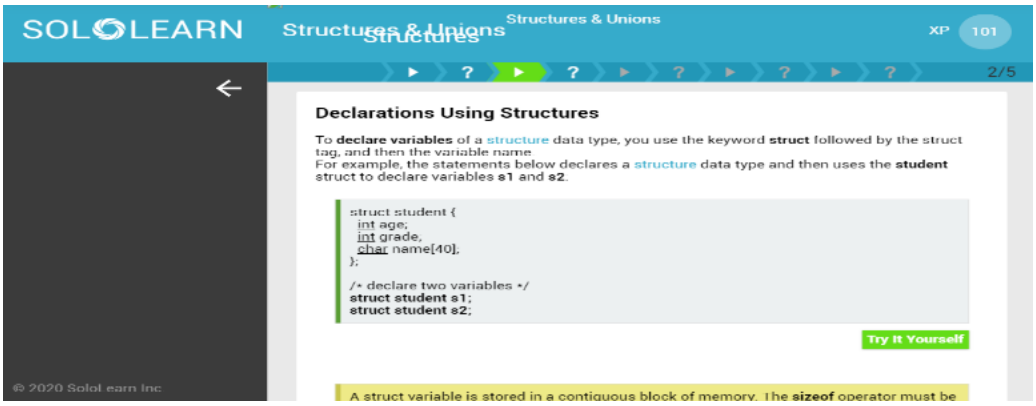
## FORENOON SESSION DETAILS

### Image of session




SOLOLEARN
Structures & Unions
Structures & Unions
Structures
XP 101
1/5
Structures
A structure is a user-defined data type that groups related variables of different data types.
A structure declaration includes the keyword struct, a structure tag for referencing the structure, and curly braces { } with a list of variable declarations called members.
For example:
struct course {
int id;
char title[40];
float hours;
};
This struct statement defines a new data type named course that has three members. Structure members can be of any data type, including basic types, strings, arrays, pointers, and even other structures, as you will learn in a later lesson.
Do not forget to put a semicolon after structure declaration.
A structure is also called a composite or aggregate data type. Some languages refer to structures as records.
© 2020 SoloLearn Inc.


SOLOLEARN
Structures & Unions
Structures & Unions
Structures
XP 101
2/5
Declarations Using Structures
To declare variables of a structure data type, you use the keyword struct followed by the struct tag, and then the variable name.
For example, the statements below declares a structure data type and then uses the student struct to declare variables s1 and s2.
struct student {
int age;
int grade;
char name[40];
};
/* declare two variables */
struct student s1;
struct student s2;
Try It Yourself
A struct variable is stored in a contiguous block of memory. The sizeof operator must be
© 2020 SoloLearn Inc.

C Programming

9/06/2020 - Friday

Strings & Function Pointers

> Strings

A string in C is an array of characters that ends with a NULL character ('\0').

A string declaration can be made in various ways, each with its own considerations. For example:

char str_name[str_len] = "string";

String point declarations such as char *str = "string";

strlen() - get length of a string
strcat() - merge two strings
strcpy() - copy one string to another
strlwr() - convert string to lowercase
strupr() - convert string to uppercase
strrev() - reverse string
strcmp() - compare two strings

String Input:

Programs can receive interactive string input.

You can use scanf() to read input according to the format specifier.

A safer alternative to gets() is fgets() which reads up to a specified no. of characters.

> String Output

String output is handled with fputs(), puts() & printf() functions.

fprintf() requires the name of the stream & a pointer to the data you want to print. The string to print to the screen, can be stdout which refers to standard output.

> String Functions

The sprintf() & sscanf() functions

A formatted string can be created with the sprintf() function. This is useful for building a string from other data types.

Another useful function is sscanf() for scanning a string for values. The function will read values from a string & store them at the corresponding variable addresses.

> The string.h library

The string library contains numerous string functions.

> Converting String to Number

Converting a string of no. characters to numeric value is a common task in C programming & is often used to process user-input data.

> Array of Strings

A two-dimensional array can be used to store related strings.

## Function Pointer

Just a pointer can point to an address in any memory location, they can also point to the start of executable code.

return_type (*func_name)(parameters)

- **Array of Function Pointers**

An array of function pointer can replace a switch statement for choosing an action, as in following

- **The void pointer**

A void pointer is used to refer to any data type in a memory. It has a declaration that looks like `void *ptr;`

- **Functions using void Pointers**

Void pointer are often used in function declarations.

`void *square(const void *);`

- **Function pointers as arguments**

A function pointer used as an argument is sometimes referred to as a callback function, because the receiving function 'calls it back'.

`void qsort(void *base, size_t num, size_t width, int (*compar)(const void *, const void *));`

## Structures & Union Structures & Unions

A structure is a user-defined data type that groups related variables of different data types.

- **Declaration Using Structures**

To declare a variables of a structure data type, you use the keyword `struct` followed by the struct tag & then the variable name.

- **Declarations Using attributes**

Additional variable can also be initialized in the declaration by listing initial values in order inside curly braces.

- **Accessing Structure Members**

You can access the members of a struct variable by using `.` (dot Operator). Just use the variable name & the member name.

- **Using typedef**

The typedef keyword creates a type definition that simplifies code & makes a program easier to read.

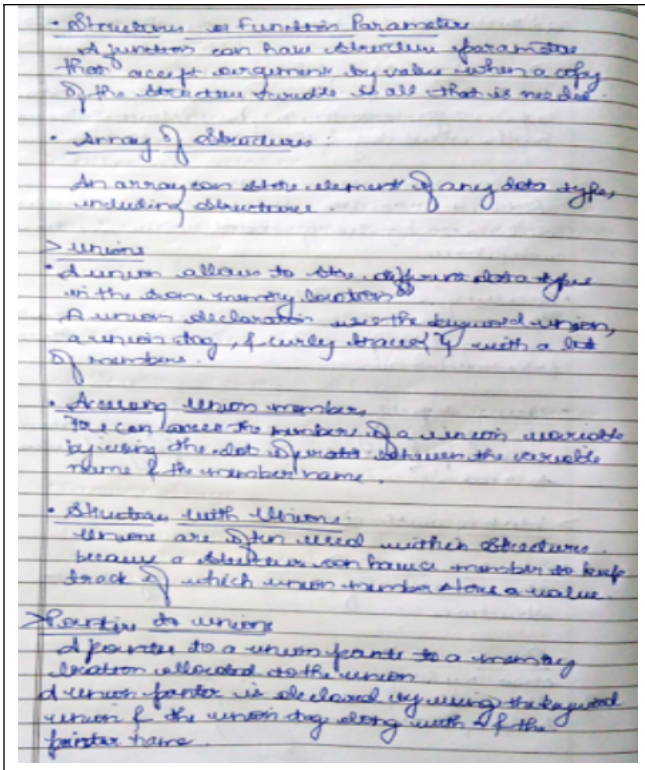
## > Working with Structures

- **Structures with Structures**

The members of a structure may also be structures.

- **Pointers to structures**

```
struct my_struct *struct_ptr;
struct_ptr = &struct_var;
struct_ptr -> struct_mem;
```



• Structures as Function Parameter

A function can have structure parameter that accept arguments by value. When a copy of the structure is made and that is passed to the function.

• Array of Structures:

An array can store the element of any data type, including structures.

> Unions

• A union allows to store different data types in the same memory location.

A union declaration uses the keyword union, a union tag, { curly braces } with a list of members.

• Accessing Union members

To access the members of a union variable by using the dot (.) operator between the variable name & the member name.

• Structure with Unions:

Unions are often used within structures because a structure can have a member to keep track of which union member store a value.

> Pointer to unions

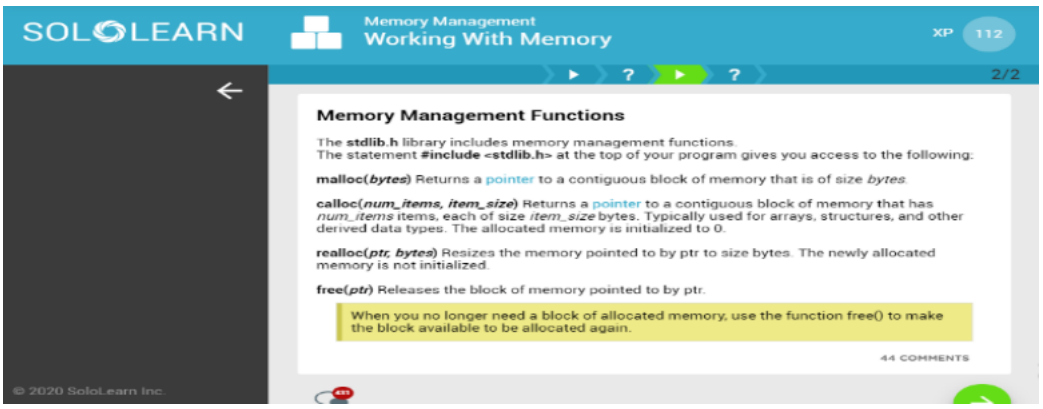
A pointer to a union points to a memory location allocated to the union.

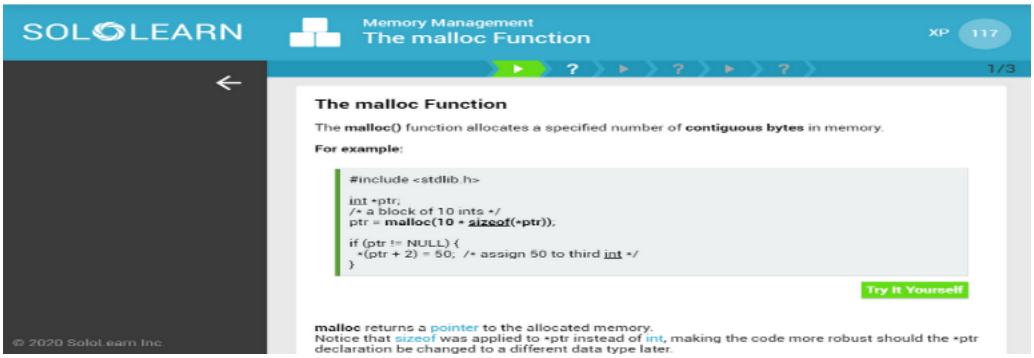
A union pointer is declared by using the keyword union & the union tag along with * of the pointer name.

| | | | |
|---|---|---|---|
| Date: | 19-06-2020 | Name: | Rajeshwari Gadagi |
| Course: | Solo Learn | USN: | 4AL17EC076 |
| Topic: | C Programming | Semester & Section: | 6 SEM & 'B' SEC |
| Github Repository: | Rajeshwari-gadagi | | |

## AFTERNOON SESSION DETAILS

### Image of session



SOLOLEARN

Memory Management
calloc and realloc

XP 122

2/2

## The realloc Function

The **realloc()** function expands a current block to include additional memory.
**For example:**

```
int *ptr;
ptr = malloc(10 * sizeof(*ptr));
if (ptr != NULL) {
  *(ptr + 2) = 50;  /* assign 50 to third int */
}
ptr = realloc(ptr, 100 * sizeof(*ptr));
*(ptr + 30) = 75;
```

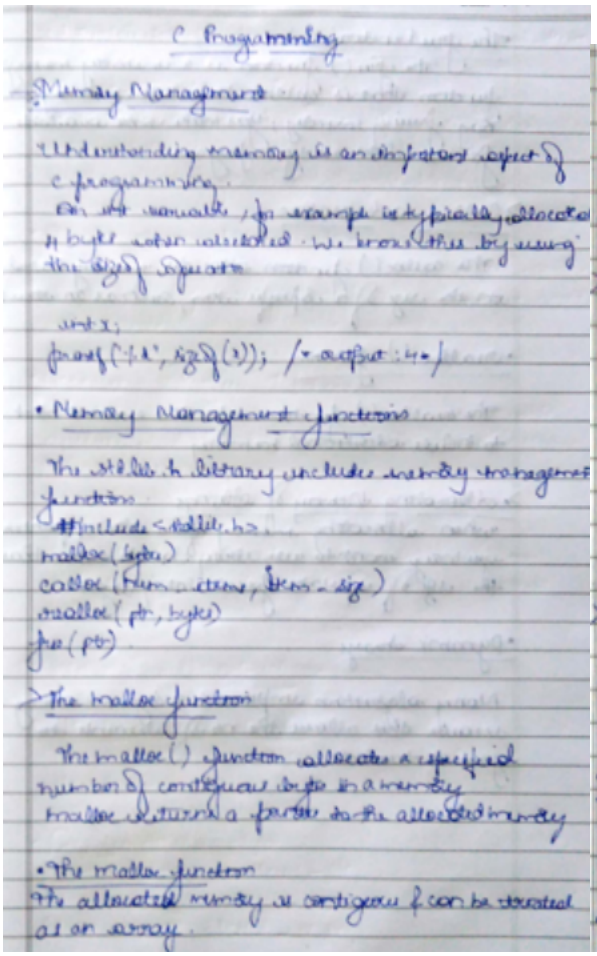
Try It Yourself

**realloc** leaves the original content in memory and expands the block to allow for more storage.

Tap **Try It Yourself** to play around with the code.

60 COMMENTS





C Programming

• Memory Management

Understanding memory is an important aspect of c programming.
On the variable, for example is typically allocated 4 byte when initialized. We know this by using the sizeof() operator.

```
int x;
printf("%d", sizeof(x)); /* output : 4 */
```

• Memory Management functions

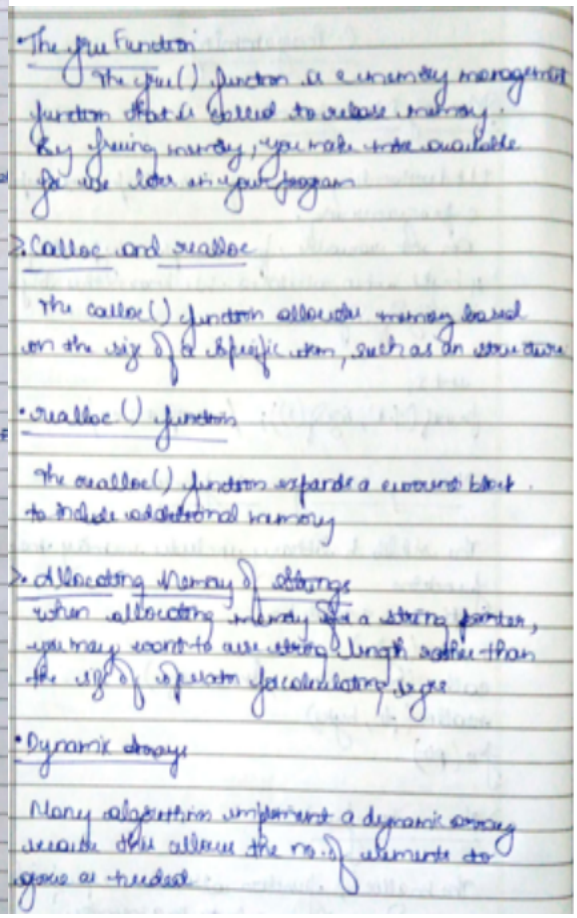
The stdlib.h library includes memory management functions

```
#include <stdlib.h>
malloc(bytes)
calloc(num_items, item_size)
realloc(ptr, bytes)
free(ptr)
```

> The malloc function

The malloc() function allocates a specified number of contiguous bytes in memory
malloc returns a pointer to the allocated memory

• The malloc function
The allocated memory is contiguous & can be treated as an array.



• The free Function

The free() function is a memory management function that is called to release memory. By freeing memory, you make it available for use later in your program.

> calloc and realloc

The calloc() function allocates memory based on the size of a specific item, such as an structure

• realloc() function

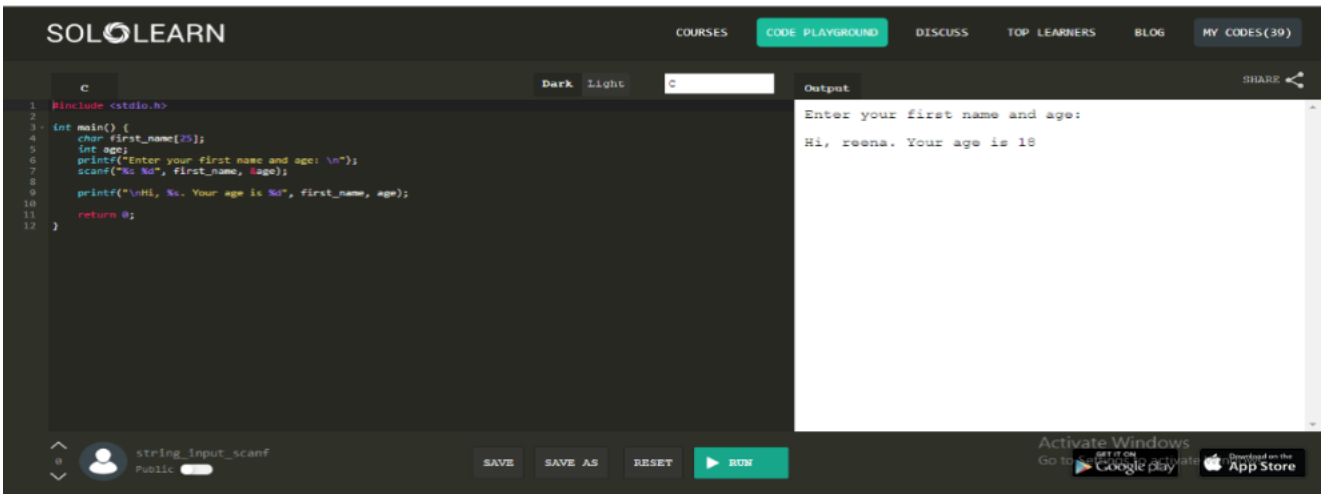
The realloc() function expands a current block to include additional memory

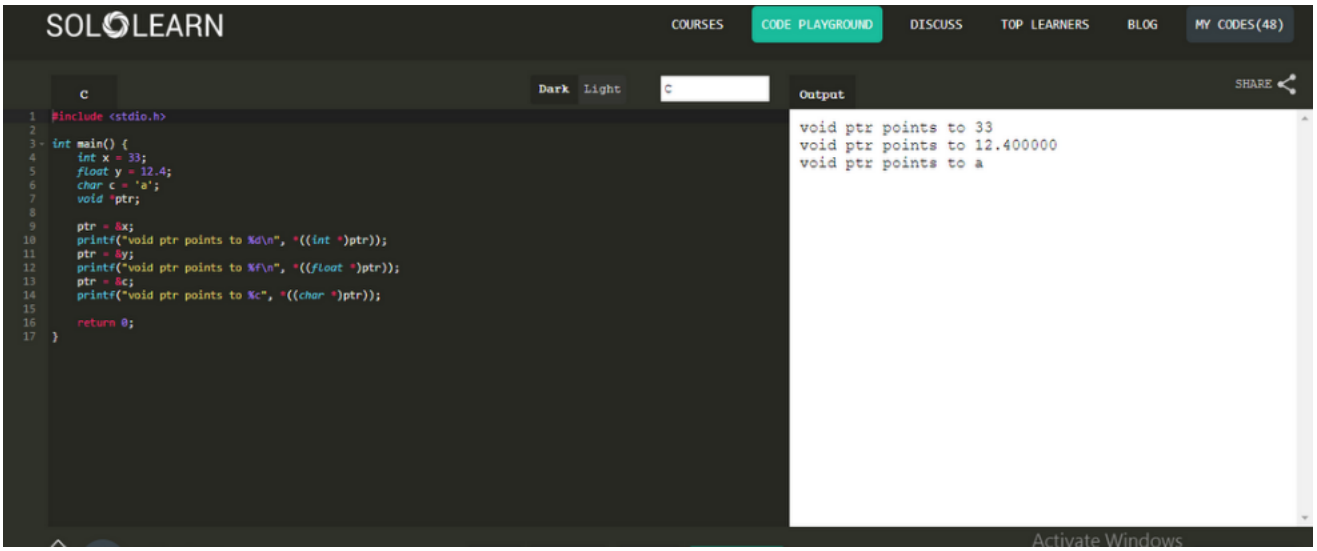
> Allocating Memory of strings

When allocating memory for a string pointer, you may want to use string length rather than the use of operator for calculating size

• Dynamic arrays

Many algorithms implement a dynamic array, which allows the no. of elements to grow as needed


SOLOLEARN
COURSES
CODE PLAYGROUND
DISCUSS
TOP LEARNERS
BLOG
MY CODES(39)
C
Dark
Light
Output
SHARE
#include <stdio.h>
int main() {
char first_name[25];
int age;
printf("Enter your first name and age: \n");
scanf("%s %d", first_name, &age);
printf("\nHi, %s. Your age is %d", first_name, age);
return 0;
}
Enter your first name and age:
Hi, reena. Your age is 18
string_input_scanf
Public
SAVE
SAVE AS
RESET
RUN
Activate Windows
Google play
App Store


SOLOLEARN
COURSES
CODE PLAYGROUND
DISCUSS
TOP LEARNERS
BLOG
MY CODES(48)
C
Dark
Light
Output
SHARE
#include <stdio.h>
int main() {
int x = 33;
float y = 12.4;
char c = 'a';
void *ptr;
ptr = &x;
printf("void ptr points to %d\n", *((int *)ptr));
ptr = &y;
printf("void ptr points to %f\n", *((float *)ptr));
ptr = &c;
printf("void ptr points to %c", *((char *)ptr));
return 0;
}
void ptr points to 33
void ptr points to 12.400000
void ptr points to a
Activate Windows