Functions

اس کو sub گروپ بھی کہتے ہیں اس کی وجہ سے پروگرام آسان ہو جاتا ہے۔

Functions rule

1) Function Decleration
2) Function Call
3) Function Defination
4) Function Return

Function Decleration

اس کے اندر وہی کرتے ہیں جو پہلے پروگرام میں کرتے ہیں کہ اس کو بتاتے ہیں ہم کو ویلیو کس ٹائپ کی چاہیے int, float وغیرہ۔ اس کے اصول یہ ہیں۔

i) Function type(Return type)

Function میں ہم نے بتانا ہوتا ہے Function کو ~~کہ~~ int, float ویلیو دینے۔ اس کو Function type کہتے ہیں۔

Function name

اس کے بعد ہم نے یہ بتانا ہے اس Function کو name کیا ہے۔

```
int name(int x, int y);
```

↓ Function name

Parameter (Argument)

```
int name(int x, int y);
```

↓ Parameter or argument

Terminal Semicolon

int name(int, int);

Terminal Semicolon

Terminal ہیں اس کے اندر یہ ہوتا ہیں

ہوتا ہیں۔

Declaration function

اس کے اندر Terminal Semicolon ہوتا ہیں

Terminal Semicolon میں Definition function

نہیں ہوتے۔

Function Definition

- i) Function type
- ii) Function name
- iii) parameter list

— Function Header

- iv) local variable Declarations
- v) Function Statements
- vi) a return Statement

— Function Body

Function Definition (اس کے آخر میں)

Function Body ہوتے اور یہ terminal semicolan

نہیں ہوتے۔

local Variable

Function Header

type function name(parameters list)

Declaration Function

int add(int a, int b);
int add(int, int);
int add(int x, int y) Definition Function

Definitions

اس کے اندر یہ ہوتا ہے کہ ہم نے parameter دیے ہیں اس کو Change کر سکتے ہیں ۔
جو Declaration function کے variable ہیں Definition function میں وہ parameter or argument ہیں ان کو Change بھی کر سکتے ہیں اور Same بھی رکھ سکتے ہیں ۔

Function Body

Header

{

variable x ⟵ اگر ہم اس کو نہ لکھیں تو کام چل سکتا ہے لیکن اگر Statement نہ دے تو program run نہیں ہوگا

Statement

return Statement

اور اگر ہم return statement نہ دے تو program چل جائے گا اس کا کوئی issue نہیں آئے گا لیکن اس statement میں last میں [illegible] نہیں

Void :

```
int message(Void);
void add(int, int);
void add(Void);
```

Void اس لیے استعمال ہوتا ہے جب
ہم نے کچھ نہیں کرنا ہوتا ہے مثلاً

int message(void);

* اس کے اندر return تو value نے والی
ہیں بیان جو void استعمال کیا ہیں
اس کا مطلب یہ ہیں ہم arguments کے
pass نہیں کرنا۔

* void add(int, int)

اس میں ہم نے کوئی ویلیو تو return نہیں کروانی
مگر اس میں argument کو pass کر دیئے گئے

* void add(void)

اس کے اندر ہم نے نہ کوئی ویلیو return کروائی
اور نہ ہی argument کو pass کروائے ہیں گئے

```
Type function name(parameter list)
  local variable declaration;
  function statement 1;
     "       "   statement 2;
  a return value;
```

int y — اس کے اندر local variables ہی ہم کو
استعمال کرتے ہیں تو program کو run کرنے کے لیے
ہوگا۔

Scanned with CamScanner

Argument type

* Formal arguments
* actual arguments

```
#include <stdio.h>
int add(int x, int y);
main()
{
    add(x, y)->function call
```

formal arguments

actual argument

جب ہم function call کرتے ہیں تو جو argument ہم دیتے ہیں اس کو actual argument کہتے ہیں اور جب جس کو یہ دیا جاتا ہے یا جس کو کال کی جاتی ہے اس کو formal argument کہتے ہیں۔

Category of function

1) Function with argument and return
2) // with no argument and return
3) // // with argument and no return
4) // // with no argument and no return

(1) Function with argument and return

int add(int x, int y)

اس کے اندر argument بھی pass کریں گے

اور return بھی کرے گا۔

```
#include <stdio.h>
int add(int x, int y);
main()
{
    int x,y; int c;
    printf("Enter your value");
    Scanf("%d%d", &x, &y);
    c = add(x,y);
    printf("th addition is %d", c);
}
int add(int a ;int b)
{
    int y;
    y = a+b;
    return y;
}
```

2) function with no arrgument and re. turn

```
#include<stdio.h>
int int add(void)
main()
 {
   int y;
  y = add(void);
printf("addition %d",y);
 }
  int add(void)
{
int a, b, c;
printf("Enter a value");
scanf("%d%d", &a, &b);
 c= a+b;
printf("four value%d", c);
;   return c;
```

Function with argument and no retu

```
#include<stdio.h>
void add(int x, int y);
main()
{
  int  a, b, c;
printf("Enter your value");
scanf("%d %d", &a, &b);
 add(x, y);
}
void add(int x, int y)
{
   int  z;
  z = x+y;
printf("%d", z);
}
```

```
#include<stdio.h>
void add(void);
main()
{
  add();
}
void add(void)
{
  int a, b, c;
printf("Enter your value");
 scanf("%d%d", &a, &b);
  c= a+b;
printf("%d", c);
 return;
}
```

Function call by value

void disp(int x);

main()

{

int a=100; اس کی real value یہ ہیں

disp(a); اس کی calculation ہونے کے

بعد x کی value

200 ہو جائے گی

void disp(int x) جو a کی values ہیں

100 ہیں وہ آ رہی ہیں اس میں کو calculation کر رہے ہیں

{

X = X + 100; اس کی calculation

جب function اس طرح call by value ہو تو اس کے اندر اس طرح ہوتا ہیں۔ جو a کی real value ہیں اس کی copy ہو رہی ہیں وہ function Definition میں جاتی ہیں اور اس کی calculation ہونے کے بعد جو ہمارے pass value ہوا اس میں change نہیں ہو جیسا کہ اوپر ہیں۔

Function: Call by reference

void disp(int *x);

main()

{

int a =100;

disp(&a);

printf("%d", a); }

disp(x)
{
*x = *x + 100; ——— real value بھی
}
اس کے اندر pointer کا اصول استعمال ہوتا
ہیں اس کو جو real value ہوتی ہیں
اس کے ساتھ جو address ہیں اس کی وجہ
سے پوری ویلو calculation کی change ہو جاتی
ہے دیکھو اس میں ہیں۔ اس میں
a کی ویلو 100 ہیں پہلے جب function Difinition
میں اس کی calculation ہوتو وہ value ہوتی جو
a کی ہیں۔ 200 جو اس کو change
reference کہتے ہیں۔

Function passing array

اس کے اندر ہم array کو pass کرتے
جب ہم array کو pass کرتے تو اس کا
Method تھوڑا سا مختلف ہوگا
جب ہم اس کو Declear کرتے تو ایسے
without array passing لکھتے ہیں
disp(int, int);
یہ نام بدل سکتے ہیں ہم نے کوئی argument کو
pass کرنا ہو