

Estruturas em C

```
struct Data
{
    int dia;
    int mes;
    int ano;
    int status;
};
```

Estruturas em C

```
| struct Data data_inicia(struct Data dt)
| {
|     dt.dia = 23;
|     dt.mes = 9;
|     dt.ano = 2017;
|     dt.status = 0;
|     return dt;
| }
```

Estruturas em C


```
struct Data data_altera(struct Data dt, int dia, int mes, int ano)
{
    dt.dia = dia;
    dt.mes = mes;
    dt.ano = ano;
    if(/*validarData()*/ true)
    {
        dt.status = 1;
    } else {
        dt.status = 0;
    }
    return dt;
}
```

Estruturas em C

```
void data_imprime(struct Data dt)
{
    if(dt.status)
    {
        //Imprime data no lcd, serial..etc
    } else {
        //mensagem de erro
    }
}
```

Estruturas em C

```
int main(void)
{
    struct Data relógio;
    relógio = data_inicia(relógio);
    data_imprime(relógio);
    relógio = data_altera(relógio, 12, 10, 2017);
    data_imprime(relógio);

    while (1)
    {
        
    }
}
```

Estruturas em C

Qual o erro cometido abaixo?

```
struct Data rtc;  
data_imprime(rtc);
```

Estruturas em C

```
//Passagem de parâmetro por CÓPIA
relogio = data_inicia(relogio);
| struct Data data_inicia(struct Data dt)
| {
|     dt.dia = 23;
|     dt.mes = 9;
|     dt.ano = 2017;
|     dt.status = 0;
|     return dt;
| }
```

Estruturas em C

```
//Passagem de parâmetro por CÓPIA
relogio = data_inicia(relogio);
| struct Data data_inicia(struct Data dt)
| {
|     dt.dia = 23;
|     dt.mes = 9;
|     dt.ano = 2017;
|     dt.status = 0;
|     return dt;
| }
```


Estruturas em C

//Passagem por referência

```
data_inicia(&relogio);
```

```
void data_inicia(struct Data * dt)
{
    dt->dia = 23;
    dt->mes = 9;
    dt->ano = 2017;
    dt->status = 0;
}
```

Estruturas em C

//Passagem por referência

```
data_inicia(&relogio);
```

```
void data_inicia(struct Data * dt)
{
    (*dt).dia = 23;
    (*dt).mes = 9;
    (*dt).ano = 2017;
    (*dt).status = 0;
}
```

Estruturas em C

//Caso a estrutura seja enviada como parâmetro
E seja tratada apenas como leitura, podemos
definir *o parâmetro como const*;

```
data_imprime(relogio);
void data_imprime(const struct Data dt)
{
    if(dt.status)
    {
        //dta.data = 10; error!!!
        //Imprime data no lcd, serial..etc
    } else {
        //mensagem de erro
    }
}
```

Como ficaria o mesmo programa em C++?

Classes em C

```
class Data
{
    private:
        char dia;
        char mes;
        short ano;
        bool status;

    public:
        Data(/* Data *this */); //Construtora
        void data_altera(/*Data *this */ int dia, int mes, int ano);
        void data_imprime(/*Data *this */) const;
};
```

Classes em C++

```
Data::Data()  
{  
    this->dia = 30;  
    this->mes = 9;  
    this->ano = 2017;  
    this->status = true;  
}
```

Classes em C++

```
void Data::data_altera(/*Data *this */ int dia, int mes, int ano)
{
    this->dia = dia;
    this->mes = mes;
    this->ano = ano;
    this->status = true;
}
```

Classes em C++

```
void Data::data_imprime(/*const Data *this */) const
{
    if(this->status)
    {
        //imprime no LCD
    } else {
        //mensagem de error!
    }
}
```


Classes em C++

```
int main(void)
{
    Data dt;
    dt.data_imprime();
    dt.data_altera(12, 10, 2017);
    dt.data_imprime();

    while (1)
    {
    }
}
```

E se tivéssemos que comparar duas variáveis **Data?** Como ficaria em C++ ?

```
Data dt1, dt2;  
if(dt1 == dt2)  
{  
    //datas iguais...  
}  
... ..
```

Podemos usar o recurso :
Sobrecarga de Operadores
(*Operators Overloading*) do
C++

Sobrecarga de Operadores do C++

```
class Data
{
    private:
        char dia;
        char mes;
        short ano;
        bool status;

    public:
        Data(/* Data *this */); //Construtora
        void data_altera(/*Data *this */ int dia, int mes, int ano);
        void data_imprime(/*Data *this */) const;
        int Compare(const Data &r2) const;
        bool operator == (const Data & r2) const;
};
```

Sobrecarga de Operadores do C++

```
int Data::Compare(const Data &r2) const
{
    if(this->ano == r2.ano)
        return (this->ano - r2.ano);
    return (this->mes != r2.mes)? this->mes - r2.mes
                                : this->dia - r2.dia;
}
```

```
bool Data::operator == (const Data & r2) const
{
    PORTD = 0;    //Necessário para poder simular no Atmel Studio
    return (this->Compare(r2)==0);
}
```

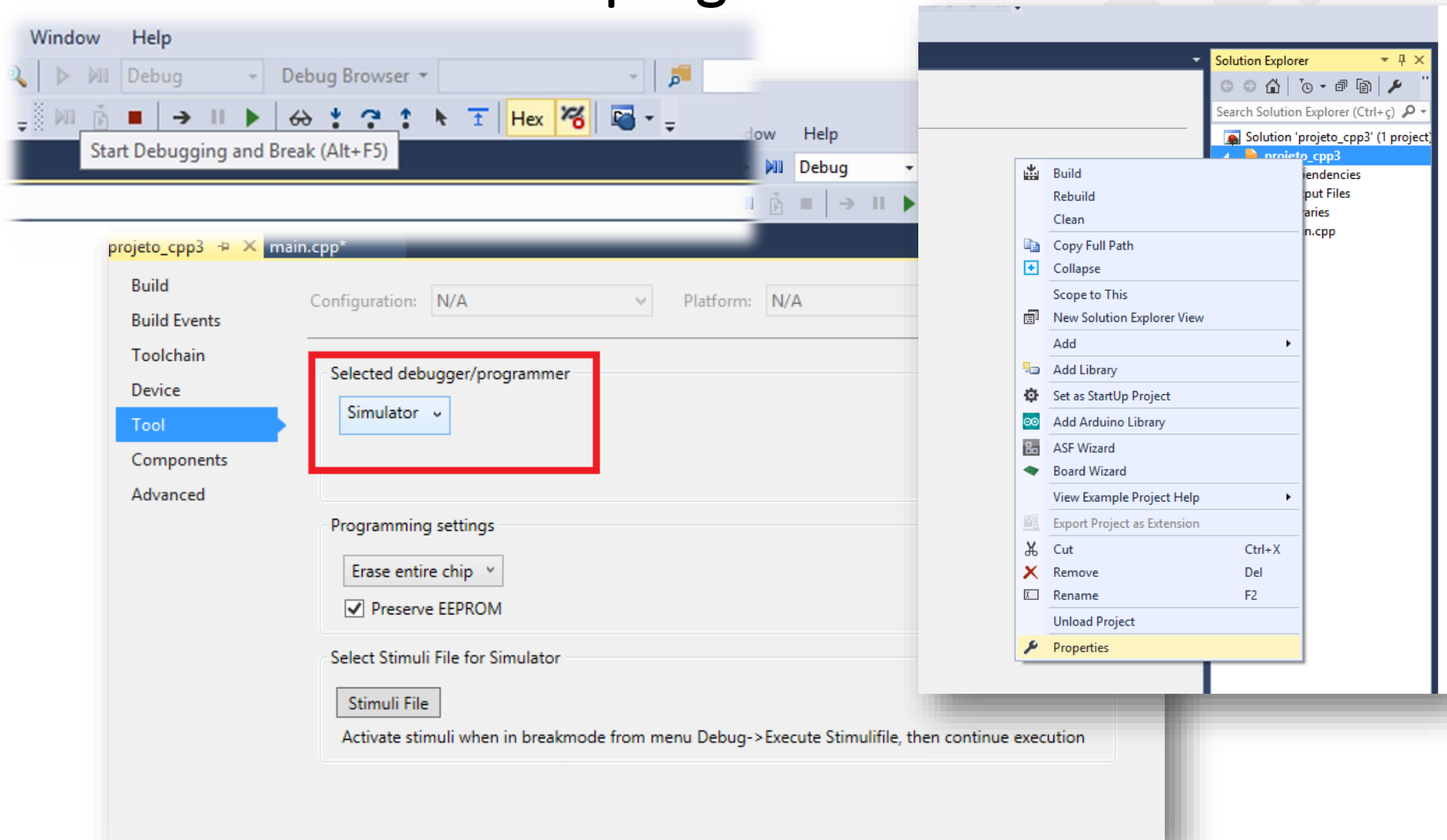
Sobrecarga de Operadores do C++

```
int main(void)
{
    Data dt;
    dt.data_imprime();
    dt.data_altera(12, 10, 2017);
    dt.data_imprime();

    Data dt1, dt2;
    if(dt1 == dt2)
    {
        //datas iguais...
    }
    while (1)
    {
    }
}
```

**Vamos simular o programa
no Atmel Studio?**

Vamos simular o programa no Atmel Studio?



O C++ suporta: Sobrecarga de funções (*Function Overloading*)

Sobrecarga de função no C++

```
class UART
{
    private:
        char txt[50];

    public:
        UART(/* UART *this */);
        void writebyte(unsigned char data);
        void writestr(char *ptr);
        void print(int i);
        void print(float f);
        void print(char* c);

};
```

Sobrecarga de função no C++

```
void UART::print(int i)
{
    sprintf(txt, "AN0 = %d \n", i);
    writestr(txt);
}

void UART::print(float f)
{
    sprintf(txt, "AN0 = %f \n", f);
    writestr(txt);
}

void UART::print(char* c)
{
    sprintf(txt, "AN0 = %c \n", c);
    writestr(txt);
}
```

Sobrecarga de função no C++

```
void UART::writestr(char *ptr)
{
    while(*ptr != '\0')
    {
        writebyte(*ptr);
        ptr++;
    }
}

void UART::writebyte(unsigned char data)
{
    while(!(UCSR0A & (1<<UDRE0)));
    UDR0 = data;
}
```

Sobrecarga de função no C++

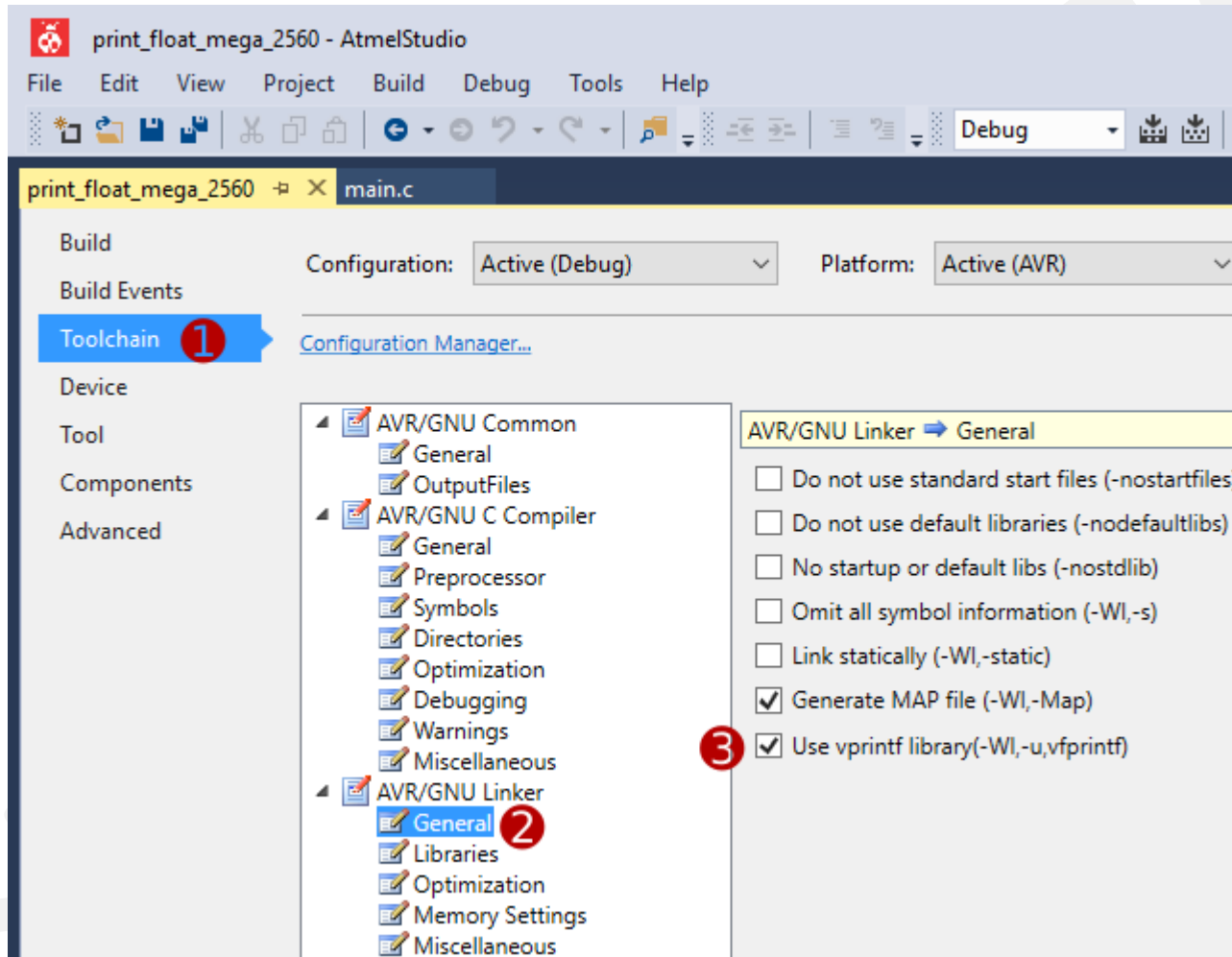
```
int main(void)
{
    int i = 10;
    float t = 5.4;
    char c = 'A';

    UART serial;
    serial.print(i);
    serial.print(t);
    serial.print(c);

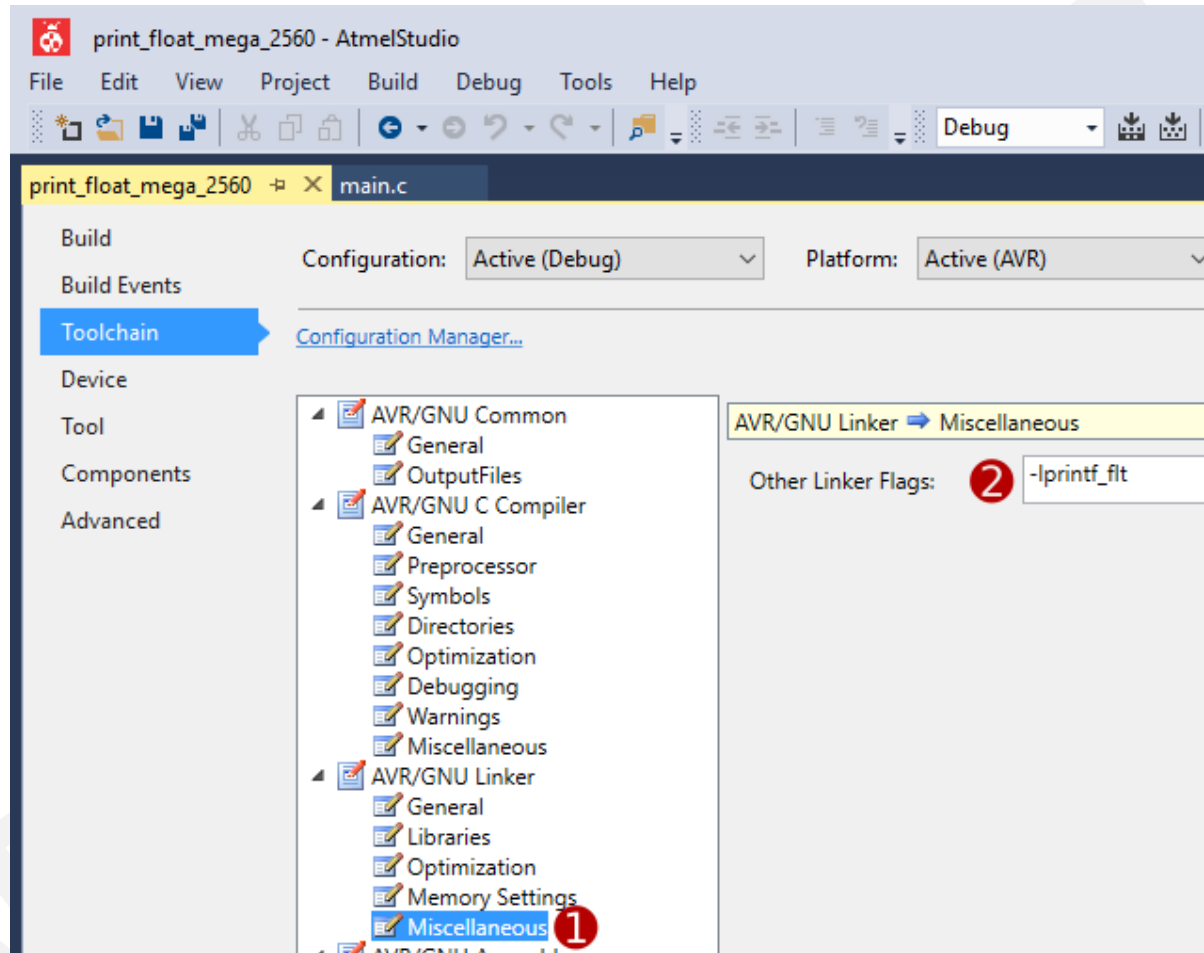
    while(1);
}
```

E aí, o *sprintf(float)*
funcionou?

Sobrecarga de função no C++



Sobrecarga de função no C++



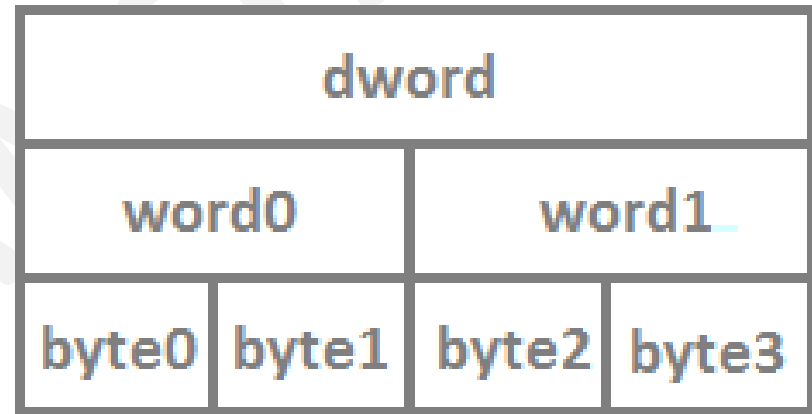
Referência: <https://startingelectronics.org/articles/atmel-AVR-8-bit/print-float-atmel-studio-7/>

Union em C

```
union Valor
{
    uint32_t dword;

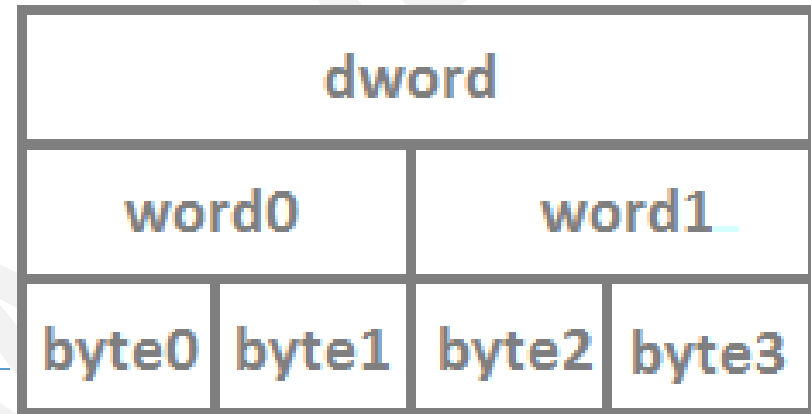
    struct
    {
        uint16_t word0;
        uint16_t word1;
    };

    struct
    {
        uint8_t byte0;
        uint8_t byte1;
        uint8_t byte2;
        uint8_t byte3;
    };
};
```



Union em C

```
struct bits {
    uint8_t bit0 : 1;
    uint8_t bit1 : 1;
    uint8_t bit2 : 1;
    uint8_t bit3 : 1;
    uint8_t bit4 : 1;
    uint8_t bit5 : 1;
    uint8_t bit6 : 1;
    uint8_t bit7 : 1;
};
```



Union em C

```

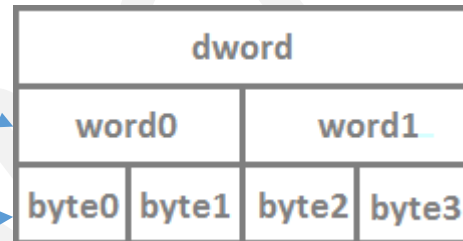
union Valor
{
    uint32_t dword;

    struct
    {
        uint16_t word0;
        uint16_t word1;
    };

    struct
    {
        uint8_t byte0;
        uint8_t byte1;
        uint8_t byte2;
        uint8_t byte3;
    };

    struct bits {
        uint8_t bit0 : 1;
        uint8_t bit1 : 1;
        uint8_t bit2 : 1;
        uint8_t bit3 : 1;
        uint8_t bit4 : 1;
        uint8_t bit5 : 1;
        uint8_t bit6 : 1;
        uint8_t bit7 : 1;
    };
};

```



Union em C

```

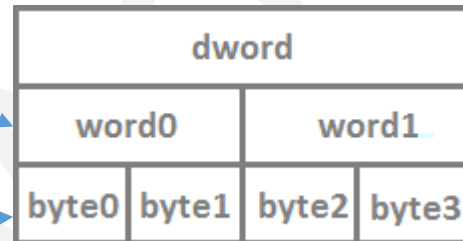
union Valor
{
    uint32_t dword;

    struct
    {
        uint16_t word0;
        uint16_t word1;
    };

    struct
    {
        uint8_t byte0;
        uint8_t byte1;
        uint8_t byte2;
        uint8_t byte3;
    };

    struct bits {
        uint8_t bit0 : 1;
        uint8_t bit1 : 1;
        uint8_t bit2 : 1;
        uint8_t bit3 : 1;
        uint8_t bit4 : 1;
        uint8_t bit5 : 1;
        uint8_t bit6 : 1;
        uint8_t bit7 : 1;
    };
};

```



Union em C

```
int main(void)
{
    char txt[20];
    volatile union Valor data;

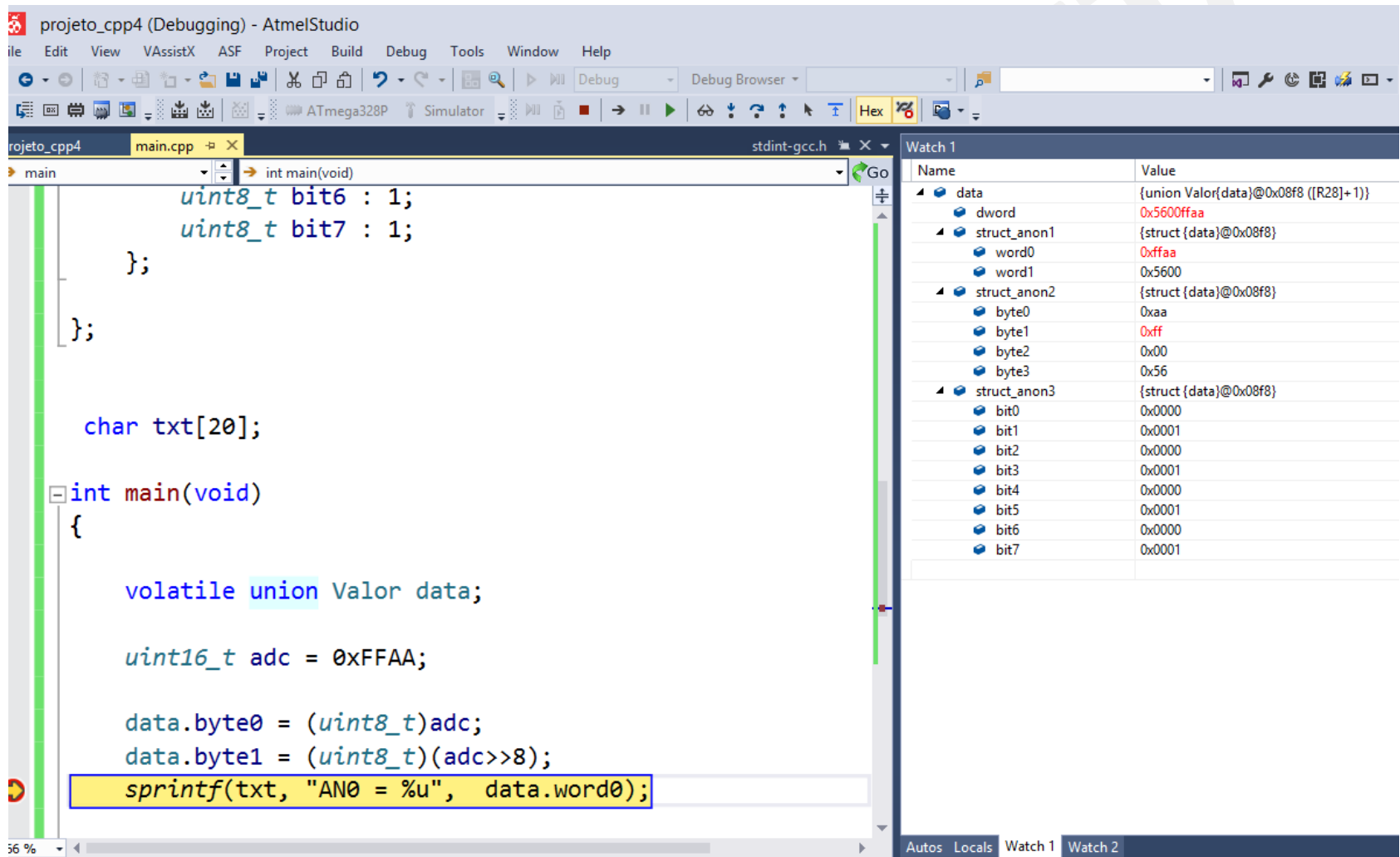
    uint16_t adc = 0xFFAA;

    data.byte0 = (uint8_t)adc;
    data.byte1 = (uint8_t)(adc>>8);
    sprintf(txt, "%u", data.word0);
    //Resultado: txt[] = {"65450"}

    data.bit0 = 1;

    while (1);
}
```

Union em C



projeto_cpp4 (Debugging) - AtmelStudio

File Edit View VAssistX ASF Project Build Debug Tools Window Help

Debug Browser

ATmega328P Simulator

Hex

main.cpp

main

```
uint8_t bit6 : 1;
uint8_t bit7 : 1;
};

};

char txt[20];

int main(void)
{
    volatile union Valor data;

    uint16_t adc = 0xFFAA;

    data.byte0 = (uint8_t)adc;
    data.byte1 = (uint8_t)(adc>>8);
    sprintf(txt, "AN0 = %u", data.word0);
}
```

Watch 1

Name	Value
data	{union Valor{data}@0x08f8 ([R28]+1)}
dword	0x5600ffaa
struct_anon1	{struct {data}@0x08f8}
word0	0xffaa
word1	0x5600
struct_anon2	{struct {data}@0x08f8}
byte0	0xaa
byte1	0xff
byte2	0x00
byte3	0x56
struct_anon3	{struct {data}@0x08f8}
bit0	0x0000
bit1	0x0001
bit2	0x0000
bit3	0x0001
bit4	0x0000
bit5	0x0001
bit6	0x0000
bit7	0x0001

56 % Autos Locals Watch 1 Watch 2

Mais um exemplo de Estrutura em C

Estruturas em C

Imagine que temos um item de supermercado. Este item tem um nome, preço e dimensão. A dimensão pode ser tanto em volume (1 litro) como peso (1 Kg). Assim, poderíamos criar a seguinte *struct*

```
struct item {
    char nome[50];
    float preco;
    float volume;
    unsigned peso;
}
```

```
struct item {
    char nome[50];
    float preco;
    union {
        float volume;
        unsigned peso;
    }
}
```


Estruturas em C

Como reduzir então a memória utilizada? Podemos declarar dentro de uma union os campos volume e peso:

```
struct item {  
    char nome[50];  
    float preco;  
    union {  
        float volume;  
        unsigned peso;  
    }  
}
```

Estruturas em C

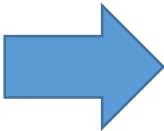
Qual o problema do programa abaixo?

```
struct item it;  
it.peso = 2;  
it.volume = 0.0f;  
printf("%u", it.peso);
```

Estruturas em C

Podemos resolver o problema adicionando uma variável “flag” a estrutura.

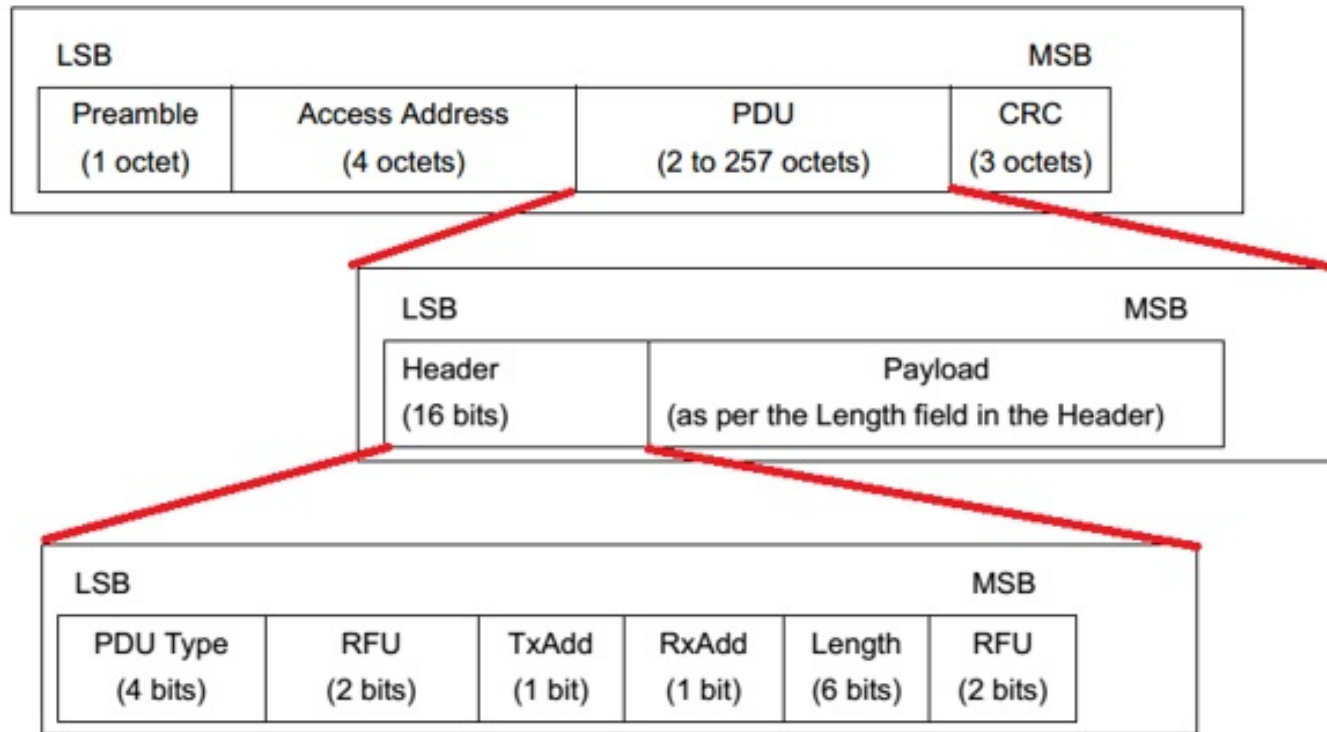
```
struct item {  
    char nome[50];  
    float preco;  
    bool porVolume;  
    union {  
        float volume;  
        unsigned peso;  
    };  
};
```



```
if ( it.porVolume ) {  
    printf("%s\t%.2f\t%.3f", it.nome, it.preco, it.volume);  
} else {  
    printf("%s\t%.2f\t%u", it.nome, it.preco, it.peso);  
}
```

Desafio

Monte uma Estrutura em C conforme a figura abaixo:



Desafio

resolução

```

struct _pdu{
    unsigned char Payload[255];
    struct{
        char PDU_TYPE:    4;
        char RFUx:        2;
        char TxAdd:       1;
        char RxAdd:       1;
        char Length:      6;
        char RFUy:        2;
    }HEADER;
};

struct{
    unsigned char Preamble;
    unsigned long Access_Address;
    struct _pdu PDU;
    unsigned long CRC;
}BLE_PACKET;

void main()
{
    BLE_PACKET.PDU.Payload[0] = 0xFF;
    BLE_PACKET.Preamble = 0b10101010;
    BLE_PACKET.Access_Address = 0b10011110;
    BLE_PACKET.PDU.HEADER.PDU_TYPE = 0b1100;
}
    
```

Obrigado!

Prof ° Fernando Simplicio

Fernando@microgenios.com.br

www.microgenios.com.br

Prof° Fernando Simplicio