pemrograman?
coding?
hacking?
hacker?

❖ **Pemrograman** :
Proses menulis, menguji dan memperbaiki (debug), dan memelihara kode yang membangun suatu program komputer. Kode ini ditulis dalam berbagai bahasa pemrograman [wiki]

❖ **Bahasa Pemrograman**
Ex/ Pascal, C, C++, Java, Python, dsb.

❖ **Kita akan mempelajari C++**

❖ **Kenapa C++?**
- Pascal terlalu *kuno*
- Kemudahan akses (fleksibilitas)
- Popularitas
- Keefisinan dan kecepatan

❖ **C vs C++**

Persamaan C dan C++ :

- Syntax yang mirip pada 2 bahasa
- Struktur kode keduanya sama
- Hampir seluruh operator dan keyword pada C juga ada pada di C++

Perbedaan C dan C++ :

- C++ ada namespace, C tidak ada
- Reference variables didukung oleh C++ bukan C
- Perbedaan dalam input dan output (scanf & printf pada C) (cin & cout pada C++) tapi C++ tetep bisa scanf & printf!
- Header file pada C adalah stdio.h sementara header file pada C++ adalah iostream.h
- C++ dianggap sebagai *upgrade*-an dari C

❖ **Bagaimana?**

- IDE (Integrated Development Environment)

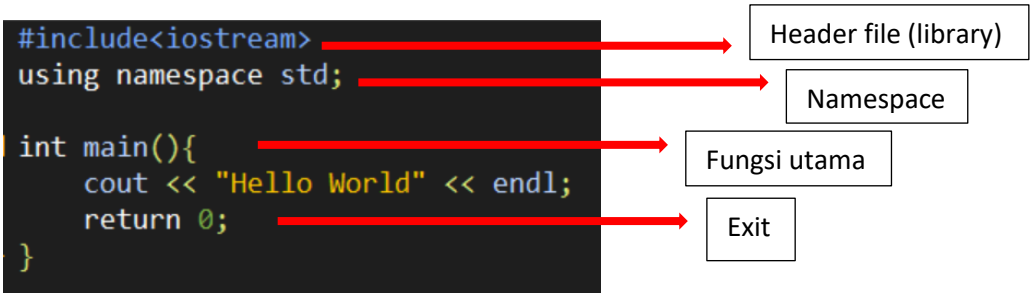
  IDE ialah aplikasi editor kode yang menyertakan compiler. Sebelum memulai menulis kode, kita butuh IDE. Contoh aplikasi IDE Bahasa C++ yang dapat digunakan :

  - ❖ DevC++
  - ❖ CodeBlocks
  - ❖ Ideone (online)

❖ **Referensi tempat belajar**

Hackerrank, toki, hackerearth, spoj, codeforces, dkk.

❖ **Contoh Program Simple**



❖ **Header File (Library)**

Di C++, semua file header mungkin atau mungkin tidak diakhiri dengan ekstensi ".h" tetapi di C, semua file header harus diakhiri dengan ekstensi ".h".
Contoh header pada C : stdio.h, string.h
Contoh header pada C++ : iostream, string.h

Kita bisa membuat header file sendiri !

❖ **Namespace**

➔ Suatu penyingkatan yang memudahkan *coder*

➔ using namespace std

= Untuk penggunaan fitur-fitur di C++ Standard Library
Sebenarnya untuk menulis cin, cout, dsb masih diperlukan “std::” di depannya.
Misalnya :

```
#include<iostream>

int main(){
    int a;
    std::cin >> a;
    std::cout << a << std::endl;
}
```

Jika program ini dilanjut dengan harus menambahkan std:: akan sangat menghabiskan waktu dan tenaga bukan?

Penulisan itu dapat disingkat menjadi :

```
#include<iostream>
using namespace std;

int main(){
    int a;
    cin >> a;
    cout << a << endl;
}
```

❖ **Fungsi Utama**

Hukumnya wajib ada fungsi utama. Kalau gaada ya program gabisa jalan guyss..

Fungsinya bertuliskan main() yang kemudian diikuti kurung kurawal, yaitu “{“ dan “}”.

Tanda “{“ menandakan awal pembuka dari fungsi main()

Pertama kali jalan yang dilihat adalah fungsi utama

- **int main() vs void main()**
  int main() mengembalikan nilai, sementara **void main()** tidak mengembalikan suatu nilai. Kalau masih bingung gapapaaa

❖ **Exit**

Untuk mengakhiri suatu program. Bisa menggunakan **exit(0)** atau **return 0**.

❖ **Identifier/pengenal**

Identifier adalah suatu nama yang biasa dipakai dalam pemrograman untuk menyatakan variabel, konstanta, tipe data, dan fungsi. Aturan untuk penulisan identifier antara lain:

1. Tidak boleh dimulai dengan karakter **non-huruf** atau symbol **underscore**, misalnya 1, 0, dsb.
2. Tidak boleh ada spasi
3. Bersifat **case-sensitive**, yang berarti mAkan dan makan memiliki makna berbeda
4. Tidak boleh menggunakan karakter-karakter ~ ! @ # $ % ^ & * ( ) + ` - = { } [ ] : " ; ' < > ? . / |
5. Tidak boleh menggunakan **reserved words** yang ada dalam C/C++.
   Reserved words? Apa itu?

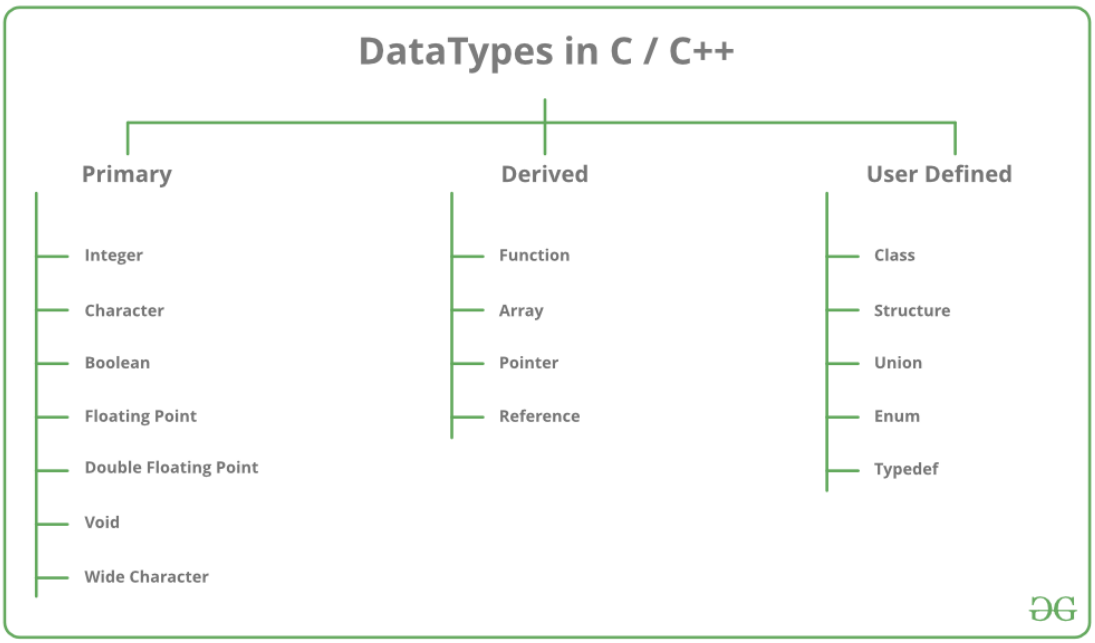
❖ **Reserved words**

Reserved Word atau Keyword merupakan kata-kata yang telah ada/didefinisikan oleh bahasa pemrograman yang bersangkutan. Kata-kata tersebut telah memiliki definisi yang sudah tetap dan tidak dapat diubah. Karena telah memiliki definisi tertentu, maka kata-kata ini tidak dapat digunakan sebagai identifier

Beberapa contoh reserved words :

auto	break	case	char
const	continue	default	delete
do	double	else	float
for	goto	if	Int
long	new	short	signed
sizeof	switch	unsigned	void

❖ **Tipe data**



Source : geeksforgeeks

Tipe Data	Memori (Byte)	Jangkauan Nilai		
		Min		Max
short	2	-2^{15}	s.d	$2^{15} - 1$
unsigned short	2 - 4	0	s.d	$2^{16} - 1$
int	4	-2^{31}	s.d	$2^{31} - 1$
unsigned int	4	0	s.d	$2^{32} - 1$
long	4	-2^{31}	s.d	$2^{31} - 1$
unsigned long	4	0	s.d	$2^{32} - 1$
long long	8	-2^{63}	s.d	$2^{63} - 1$
unisgned long long	8	0	s.d	$2^{64} - 1$

Tipe Data	Memori (Byte)	Jangkauan Nilai
float	4	$\pm 3.4 \times 10^{\pm 38}$ (estimasi)
double	8	$\pm 1.7 \times 10^{\pm 308}$ (estimasi)

Tipe Data	Memori (Byte)	Jangkauan Nilai
char	1	-2^7 s.d $2^7 - 1$
unsigned char	1	0 s.d $2^8 - 1$

Jenis Literal	Contoh	Tipe Default
Bilangan bulat	`10, 0, -1 dll.`	int
Bilangan real (floating)	`202.15, 33.24, -12.45 dll.`	double
Karakter	`'A', '1', '#'`	char
String	`"Hai"`	const char[4]

```cpp
#include<iostream>
using namespace std;

int main()
{
    cout << "Size of char : " << sizeof(char) << " byte" << endl;
    cout << "Size of int : " << sizeof(int) << " bytes" << endl;
    return 0;
}
```

❖ **Variabel**

Variabel adalah suatu tempat yang digunakan untuk menampung data atau nilai pada memori yang mempunyai nilai yang dapat berubah–ubah selama proses program. Dalam bahasa C++, variabel menyimpan data/nilai dengan tipe data tertentu. Seperti halnya variabel yang menyimpan suatu angka yang termasuk bilangan bulat (integer).

Bagaimana cara **deklarasi variabel**?

<tipe_data> <identifier>

Sebagai contoh :

```cpp
int makanan;
char nama;
float nilai;
```

Bagaimana cara **memberi nilai pada variabel**?

```cpp
int makanan = 1;
char nama;
nama = 'a';
```

❖ **Konstanta**

Konstanta adalah suatu nilai yang tidak dapat diubah atuapun berubah selama program berjalan. Bagaimana cara membuatnya?

Cara **pertama** : Tambahkan **const** di depan saat pendefinisian variable

const <tipe_data> <nama_variabel> = <nilainya>;

Contoh :

```cpp
const int nilai_tetap = 100;
nilai_tetap = 23; //error...
```

Cara **kedua** : Menggunakan prepocessor directive **define**

#define <nama_variabel> <nilainya>

Contoh :

```cpp
#define pi 3.14
```

❖ **Komentar**

- **Komentar Single-Line**
  Ditandai dengan "**//**". Kalimat atau statement setelah "**//**" tidak akan dijalankan oleh compiler (hanya berlaku untuk satu kalimat saja).
- **Komentar Multi-Line**

Jika menginginkan komentar dengan banyak baris dapat dilakukan dengan dimulai symbol "**/***" dan diakhiri "***/**"

```
//ini termasuk komentar single line

/*  kalau ini multi line
    bisa lebih dari 1 baris
    iya kan
*/
```

- **Whitespace**
  Tidak ada kode yang tertulis. Whitespace dapat berupa 3 macam, yaitu **space**, **tab**, dan **new line**.
- **Statement**
  Sebagian besar statement diakhir dengan **semicolon** (**;**)
- **Escape sequence**

Escape sequence	Karakter
\b	Backspace
\f	Form feed
\n	Newline
\r	Return
\t	Tab horizontal
\v	Tab vertical
\\	Backslash
\'	Tanda petik
\"	Tanda petik dua
\?	Tanda tanya
\0	Karakter null

- **Operator**
  - **Operator Aritmatika**
    Operasi aritmatika disini layaknya operasi yang ada di matematika seperti penjumlahan, pengurangan, perkalian, dsb.

Simbol	Keterangan	Contoh
+	Penjumlahan	A + B
-	Pengurangan	A – B
*	Perkalian	A * B

/	Pembagian	A / B
%	Sisa bagi (modulo)	A % B

- **Operator Increment dan Decrement**

  Operator increment dari kata “increase” yang berarti menambahkan atau bertambah, maka operator ini digunakan untuk menambahkan suatu nilai dengan symbol operator “++”.

  Operator decrement dari kata “decrease” yang berarti mengurangi atau dikurangi, maka operator ini digunakan untuk mengurangi suatu nilai dengan symbol operator “--".

  Terdapat 2 cara untuk menggunakan operator ini.

  1. **Prefix**

     Yaitu dengan meletakkan operator di **depan**, misalnya ++a dan --b
  2. **Postfix**

     Yaitu dengan meletakkan operator di **belakang,** misalnya a++ dan b--

- **Operator Relasional**

Simbol	Keterangan	Contoh
==	Sama dengan	5 == 1 (false)
!=	Tidak sama dengan	2 != 3 (true)
>	Lebih dari	4 > 2 (true)
>=	Lebih dari sama dengan	1 >= 3 (false)
<	Kurang dari	1 < 3 (true)
<=	Kurang dari sama dengan	3 <= 3 (true)

- **Operator Logika**

Simbol	Keterangan	Contoh
!	Logical **not**	!1 = 0
&&	Logical **and**	1 && 1 = 1
||	Logical **or**	1 || 1 = 1

- **Operator Bitwise**

Simbol	Keterangan
&	Bitwise and
|	Bitwise or
^	Bitwise xor
~	Bitwise complement
<<	Bitwise shift left
>>	Bitwise shift right

- **Operator Gabungan**

Simbol	Contoh	Ekivalen
+=	A += B	A = A + B
-=	A -= B	A = A – B
*=	A *= B	A = A * B

/=	A /= B	A = A / B
%=	A %= B	A = A % B
|=	A |= B	A = A | B
^=	A ^= B	A = A ^ B
>>=	A >>= B	A = A >> B
<<=	A <<= B	A = A << B

- **Operator Lain**
  Operator **sizeof()**, **address-of(&)**, **dereference(*)**, **kondisional(?:)**, **koma(,)**, dsb.

Latihan Soal

1. Buat program untuk menjumlahkan 5 dan 3
2. Buat program untuk menjumlahkan 2 angka sesuai masukan **user**
3. Buat program untuk menyimpan nama kamu dan menampilkannya Kembali
4. Buat program untuk menghitung keliling persegi panjang jika diketahui panjang dan lebarnya (input sesuai **user**)
5. Kerjakan tlx.toki.id bab pemrograman dasar sub bab 1-4
6. Kerjakan hackerrank
7. Buat program dengan menerima 3 angka lalu dijumlahkan dan dicari sisa bagi nya Ketika dibagi 3
8. Buat program untuk mencari jarak dari 2 titik (masukan user ada 4, yaitu x1, x2, y1, y2)
9. Input berupa jam menit detik, output detik
10. Mencari luas segitiga sembarang
11. Input 5 angka, mencari digit satuan dari penjumlahan 5 angka tsb